Server Installation

- LAMP Installation (MySQL, Apache, PHP)
- Pi-hole installieren
- Samba Netzwerkfreigabe erstellen
- Linux automatisch aktualisieren
- HTTPS Reverse Proxy mit Nginx konfigurieren
- Nginx Reverse Proxy - WebSocket verliert die Verbindung
- Apache2 SSL Virtualhost SSL Konfiguration

LAMP Installation (MySQL, Apache, PHP)

Einleitung

In dieser Anleitung beschreibe ich kurz wie wir einen **LAMP Webserver** installieren.
Ein **LAMP Webserver** ist eine Installationskombination von **Apache, MySQL / MariaDB** und **PHP** .

Voraussetzungen

Um diese Installation durchfÃ¼hren zu kÃ¶nnen sind folgende Voraussetzungen gegeben:

- Zugriff auf die Konsole Ã¼ber SSH / Telnet / Lokal
- Root Rechte / sudo Rechte
- Debian 11
- Internet Anbindung des Servers

Apache Webserver Installation

Um den Apache Webserver zu installieren, aktualisierst du zuerst die Paketquellen und installierst ggf. Updates und entfernst nicht mehr benÃ¶tigte Pakete.

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

Als nÃ¤chstes installierst du den Apache2 Webserver.

```
sudo apt install apache2 -y
```

Du kannst jetzt die Installation mit dem Befehl Ã¼berprÃ¼fen.

> Wenn du eine UFW Firewall mit installiert und aktiviert hast, musst du die Ports auf der Firewall freigeben!

```
sudo ufw allow 80/tcp && \
sudo ufw allow 443/tcp && \
```

```
sudo ufw reload
```

MariaDB Datenbank Installation

Um jetzt die MariaDB Datenbank zu installieren aktualisierst du zuerst wieder die Paket Quellen, installierst Updates und entfernst nicht mehr benÃ¶tigte Pakete.

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

Als nÃ¤chstes installierst du jetzt die MariaDB Datenbank.

```
sudo apt install mariadb-server -y
```

Und jetzt fÃ¼hren wir ein Skript aus. Dieses lÃ¤sst uns Sicherheitseinstellungen fÃ¼r unseren Datenbank Server einstellen. Dieses Ã¶ffnen wir mit folgendem Befehl

```
mysql_secure_installation
```

Hier werden einige Dinge abgefragt. Diese Einstellungen werden wir jetzt gemeinsam setzen.

- Zuerst Enter drÃ¼cken (Wir haben bisher kein root Kennwort gesetzt fÃ¼r den Datenbank Benutzer)
- Debian 11: Die Unix Authentifizierung lehnen wir mit **n** ab.
- Jetzt **Y** eingeben. Wir setzen nun ein Kennwort fÃ¼r den **Root** Datenbank Benutzer.
- Als nÃ¤chstes **Y** eingeben. Wir wollen alle unbekannten Benutzer lÃ¶schen.
- Dann geben wir wieder **Y** ein. Wir unterbinden damit eine Anmeldung des **Root Benutzers** auÃŸerhalb unseres Servers.
- Und wieder geben wir **Y** ein. Damit wird die Test Datenbank und die Rechte dorthin gelÃ¶scht.
- Als letztes geben wir wieder ein **Y** ein. Damit werden die Berechtigungen einmal neu geladen.

Nun kannst du dich mit folgendem Befehl auf dem SQL Server einloggen. Du wirst nach der Eingabe nach dem Kennwort des Root Benutzers gefragt. Sobald du dieses eingegeben hast, kannst du SQL Befehle absetzen.

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

> Wenn kein Passwort fÃ¼r Root angegeben wurde, loggt sich der Benutzer automatisch auf dem Server ein.

Installation von PHP 7.4

Um unsere Installation abzuschlieÃŸen, installieren wir jetzt PHP7.4

PHP ist eine Serverseitige Programmiersprache. Damit kÃ¶nnen Befehle direkt auf dem Server ausgefÃ¼hrt werden, z.B. werden Datenbank Abfragen hÃ¤ufig Ã¼ber PHP durchgefÃ¼hrt.

```
sudo apt install php7.4 php7.4-cli php7.4-common php7.4-curl php7.4-gd php7.4-intl php7.4-json php7.4-mbstring php7.4-mysql php7.4-opcache php7.4-readline php7.4-xml php7.4-xsl php7.4-zip php7.4-bz2 libapache2-mod-php7.4 -y
```

Du hast jetzt erfolgreich PHP 7.4 mit Modulen installiert!

Installation von PHP 8.0

Um unsere Installation abzuschlieÃŸen, installieren wir jetzt PHP7.4

PHP ist eine Serverseitige Programmiersprache. Damit kÃ¶nnen Befehle direkt auf dem Server ausgefÃ¼hrt werden, z.B. werden Datenbank Abfragen hÃ¤ufig Ã¼ber PHP durchgefÃ¼hrt.

```
sudo apt install php8.0 php8.0-cli php8.0-common php8.0-curl php8.0-gd php8.0-intl php8.0-mbstring php8.0-mysql php8.0-opcache php8.0-readline php8.0-xml php8.0-xsl php8.0-zip php8.0-bz2 libapache2-mod-php8.0 -y
```

Du hast jetzt erfolgreich PHP 8.0 mit Modulen installiert!

Optional: Installation von phpMyAdmin

Du kannst Optional auch phpMyAdmin installieren um deine Datenbank Ã¼ber eine WeboberflÃ¤che zu verwalten.

Als erstes wechselst du in das Verzeichnis in dem du phpMyAdmin ablegen mÃ¶chtest.

```
cd /var/www
```

Nun lÃ¤dst du das Verpackte Archiv mit den Dateien fÃ¼r phpMyAdmin herunter.

```
wget https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-all-languages.zip -O phpmyadmin.zip
```

Als nÃ¤chstes entpackst du das ZIP Archiv und lÃ¶schst das alte Verzeichnis.

```
sudo unzip phpmyadmin.zip && sudo rm phpmyadmin.zip
```

Nun verÃ¤nderst du den Namen des Ordners in phpMyAdmin.

```
mv phpMyAdmin-*-all-languages phpmyadmin
```

Jetzt verÃ¤nderst du die Berechtigungen auf das Verzeichnis.

```
sudo chmod -R 0755 phpmyadmin
```

Und damit wir dann Ã¼ber den Web Browser auf phpMyAdmin zugreifen kÃ¶nnen, erstellen wir eine Konfigurationsdatei fÃ¼r den Apache2 Webserver.

```
sudo nano /etc/apache2/conf-available/phpmyadmin.conf
```

Dort fÃ¼gst du folgende Konfiguration ein.

```
Alias /phpmyadmin /var/www/phpmyadmin

<Directory /var/www/phpmyadmin>
    Options SymLinksIfOwnerMatch
    DirectoryIndex index.php
</Directory>

<Directory /var/www/phpmyadmin/templates>
    Require all denied
</Directory>
<Directory /var/www/phpmyadmin/libraries>
    Require all denied
</Directory>
<Directory /var/www/phpmyadmin/setup/lib>
    Require all denied
</Directory>
```

Sobald du mit der Tastenkombination **STRG + X** und danach **Y** die Datei gespeichert hast, aktivieren wir jetzt die Konfigurationsdatei.

```
sudo a2enconf phpmyadmin && sudo systemctl reload apache2
```

Um die Installation abzuschlieÃŸen, erstellen wir temporÃ¤res Verzeichnis im **phpMyAdmin Verzeichnis** und vergeben fÃ¼r dieses die entsprechende Berechtigung.

```
sudo mkdir /var/www/phpmyadmin/tmp/ && sudo chown -R www-data:www-data /var/www/phpmyadmin/tmp/
```

Du kannst dich nun mit den entsprechenden Datenbank Benutzern anmelden. Wenn sich **phpMyAdmin** auf dem selben Server wie die Datenbank befindet, musst du keinen Server

angeben und du kannst dich mit einem Benutzer anmelden mit dem Host **localhost**. Kannst dich also auch als **root** anmelden.

Pi-hole installieren

Einleitung

Pi-hole ist ein kleiner Ad Blocker der als DNS Server arbeitet. Alle Anfragen die Ã¼blicherweise an einen Google DNS / Cloudflare DNS getÃ¤tigt werden, werden Ã¼ber das Pi-hole gesteuert. Dieser filtert die Antworten von Google und Co. nach EintrÃ¤gen die als Werbung markiert sind. So ist es mÃ¶glich das du im gesamten Netzwerk weniger bis keine Werbung mehr hast.

Installation

Es gibt 4 Wege Pi-hole zu installieren, alle werden hier beschrieben. Du musst dem Pi-hole eine **Statische IP-Adresse** geben damit dieser arbeiten kann. Du trÃ¤gst spÃ¤ter beim DHCP Server die IP-Adresse des Pi-hole's an. Wenn sich ein Client eine IP-Adresse zieht, erhÃ¤lt er zugleich die IP-Adresse des DNS Servers und alle Anfragen werden dann Ã¼ber das Pi-hole gesteuert.

Automatische Installation

Wenn du Pi-hole sich automatisch installieren lassen mÃ¶chtest, musst du nur den unten stehenden Befehl verwenden.

```
curl -sSL https://install.pi-hole.net | bash
```

Es wird hier der eigentliche Pi-hole Dienst installiert sowie ein leichtgewichtiger **lighttpd** Webserver installiert. Bei der Installation wirst du unter anderem auch gefragt ob ein Web Server erwÃ¼nscht ist.

Am Ende der Installation wird dir das Administrator Kennwort angezeigt, mit diesem meldest du dich im Web Interface an um Konfigurationen vorzunehmen.

Repository klonen und Skript ausfÃ¼hren

Als zweiten Weg kannst du das Repository von Github klonen und das entsprechende Installationsskript ausfÃ¼hren.

```
git clone --depth 1 https://github.com/pi-hole/pi-hole.git Pi-hole
cd "Pi-hole/automated install/"
sudo bash basic-install.sh
```

Installer herunterladen und ausfÃ¼hren

Wahlweise kannst du auch den dritten Weg wÃ¤hlen und den Installer herunterladen und ausfÃ¼hren um den Installationsprozess zu starten.

```
wget -O basic-install.sh https://install.pi-hole.net
sudo bash basic-install.sh
```

Installation Ã¼ber Docker

Als letzte MÃ¶glichkeit kannst du auch zu einer Installation Ã¼ber Docker tendieren. Die **docker-compose.yml** Datei findest du unten stehend.

```
version: "3"

services:
  pihole:
    container_name: server_pihole
    image: pihole/pihole:latest
    ports:
      - "53:53/tcp"
      - "53:53/udp"
      - "67:67/udp"
      - "80:80/tcp"
    environment:
      TZ: 'Europe/Berlin'
      WEBPASSWORD: 'Pa$$w0rd'
    volumes:
      - './etc-pihole:/etc/pihole'
      - './etc-dnsmasq.d:/etc/dnsmasq.d'
    cap_add:
      - NET_ADMIN
    restart: unless-stopped
```

Pi-hole CLI Befehle

Pi-hole Befehle	
Verwendung	**Befehl**
Grundbefehl	pihole
Gravity aktualsieren	pihole updateGravity
Pi-hole aktualisieren	pihole updatePihole
Pi-hole Status einsehen	pihole status
Pi-hole Version einsehen	pihole version
Administrator Kennwort ändern	pihole -a -p

Samba Netzwerkfreigabe erstellen

Einleitung

Du kannst mit Samba einen Server erstellen, auf dem du deine Dokumente in einem Netzwerkfreigabe Ordner ablegen kannst. Diesen kannst du dann unter Linux, Windows, Mac OS integrieren und so von jedem GerÃ¤t Netzwerkweit auf deine Dokumente zugreifen.

> **Achtung:** Samba 1.0 zÃ¤hlt als veraltet und sollte daher nur in lokalen abgesicherten Netzwerken installiert werden.

Installation

Um die Installation durchfÃ¼hren zu kÃ¶nnen, gibt es folgende Voraussetzungen:

- Debian 10 / 11
- root oder sudo Rechte
- Konsolenzugriff per SSH / Telnet / Lokal
- Internetanbindung des Servers

Zuerst installieren wir das Paket **samba**

```
sudo apt-get install samba
```

Als zweiten Schritt sichern wir die derzeitige Samba Konfiguration. Dieses Backup dient zum eventuellen ZurÃ¼ckspielen auf den Ursprungszustand.

```
sudo mv /etc/samba/smb.conf /etc/samba/smb.backup
```

Und nun konfigurieren wir den Samba Server. Du erstellst und Ã¶ffnest die neue Konfigurationsdatei im nÃ¤chsten Schritt.

```
sudo nano /etc/samba/smb.conf
```

Dort fÃ¼gst du die Konfiguration ein und speicherst die Datei mit der Tastenkombination `STRG + X` und danach `Y` .

```
[global]
workgroup = smb
security = user
map to guest = Bad Password

[homes]
comment = Home Directories
browsable = no
read only = no
create mode = 0750

[share]
path = /var/share/
public = yes
writable = yes
comment = smb share
printable = no
guest ok = yes
```

In der Konfigurationsdatei kannst du dann noch den Pfad zur Dateiablage verÃ¤ndern oder auch den Namen der Freigabe von **share** auf einen anderen beliebigen setzen.

> **Info:** Du verbindest das Netzlaufwerk dann Ã¼ber den UNC Namen mit dem Freigabe Namen dahinter.
> **Beispiel:** \\192.168.1.13\share

Um dann Daten abzulegen muss der Ordner ggf. erst erstellt werden und dann mit Schreibe und Lese Berechtigungen fÃ¼r Public versehen werden.

Die Berechtigungen fÃ¼r die Benutzer werden dann Ã¼ber die Samba Freigabe gesteuert.

```
sudo mkdir /var/share sudo chmod -R 777 /var/share
```

Und als letztes starten wir den Samba Service neu. Samba liest dann die neue Konfigurationsdatei ein, und die Freigabe ist dann erreichbar.

```
sudo systemctl restart smbd.service
```

Du kannst den Status des Samba Service auch Ã¼berprÃ¼fen. Setze dazu den folgenden Befehl ab:

```
sudo systemctl status smbd.service
```

Linux automatisch aktualisieren

Einleitung

In diesem Beitrag gehe ich drauf ein, wie wir mithilfe des Paketes **cron-apt** unseren Debian Server automatisch aktualisieren. Damit kÃ¶nnen wir sicherstellen, dass wenn wir im Urlaub oder nicht anwesend sind, unser Server immer auf dem aktuellen Stand ist.

Installation des Paketes

Dazu mÃ¼ssen wir zuerst die **Paketquellen aktualisieren** und gegebenenfalls **Updates installieren**, wenn diese vorhanden sind.

```
sudo apt update && sudo apt upgrade -y
```

Im nÃ¤chsten Schritt installieren wir jetzt das Paket **cron-apt**.

Dazu geben wir folgenden Befehl in die Konsole ein.

```
sudo apt install cron-apt -y
```

Konfiguration von cron-apt

Um unsere Updates automatisch installieren zu lassen, mÃ¼ssen wir **cron-apt** jetzt nur noch konfigurieren. Damit wird dann das Skript jeden Morgen um **4 Uhr Morgens** gestartet, und spielt die Updates ein. *(Die Uhrzeit kann noch geÃ¤ndert werden)*.

Wir Ã¶ffnen zuerst die Standardkonfiguration. Wir Ã¶ffnen die Datei **3-download** mit einem Editor unserer Wahl. Ich verwende hier *nano*.

```
sudo nano /etc/cron-apt/action.d/3-download
```

In dieser Datei befindet sich schon ein Befehl. Dieser Befehl wird automatisch ausgefÃ¼hrt, wenn, cron-apt gestartet wird. Dies geschieht dann automatisch, wenn die festgelegte Uhrzeit erreicht wird, oder wenn wir den Befehl `sudo cron-apt -s` eingeben.

Die Datei sollte folgenden Inhalt haben:

```
autoclean -y
dist-upgrade -d -y -o APT::Get::Show-Upgraded=true
```

Bei diesem Befehl werden die allgemeinen Updates nur heruntergeladen, aber nicht **installiert!** Wenn wir mÃ¶chten, dass die Updates automatisch installiert werden, mÃ¼ssen wir nur den Parameter **-d**entfernen. Dieser gibt an, dass die Updates nur heruntergeladen werden, und wir die Installation selbst in die Hand nehmen mÃ¼ssen.

Wenn die Updates automatisch installiert werden sollen, und man darÃ¼ber wegsieht, dass es dann zu Problemen kommen kann, durch beispielsweise zurÃ¼ckgezogene Pakete, kann man den nachstehenden Befehl anstelle des vorhandenen verwenden.

```
autoclean -y
dist-upgrade -y -o APT::Get::Show-Upgraded=true
```

Wenn wir beispielsweise wollen, dass die normalen Updates nicht automatisch installiert werden, aber Security Updates automatisch installiert werden sollen, kÃ¶nnen wir die DurchlÃ¤ufe durch eigene Skripte anpassen. Dazu mÃ¼ssen wir nur in demselben Verzeichnis eine Datei mit einer fortlaufenden Nummer und einer Beschreibung erstellen.

```
sudo nano /etc/cron-apt/action.d/10-securityupdates
```

In der Datei fÃ¼gen wir folgenden Inhalt ein:

```
upgrade -y -o APT::Get::Show-Upgraded=true
```

Damit jetzt unsere Datei auch verwendet wird, wenn **cron-apt** startet, mÃ¼ssen wir noch eine Konfigurationsdatei anlegen. Dazu legen wir wieder eine Datei mit dem Namen unser vorherigen angelegten Datei an. Dabei verÃ¤ndert sich nur der Ordner, in dem die Datei angelegt wird.

```
sudo nano /etc/cron-apt/config.d/10-securityupdates
```

Dort fÃ¼gen wir folgenden Code ein. Dabei mÃ¼ssen wir aber die entsprechenden Pfade zu den Paketquellen Listen angeben.

```
OPTIONS="-q -o Dir::Etc::SourceList=/etc/apt/sources.list.d/security.list -o Dir::Etc::SourceParts=\"/dev/null\""
```

AusfÃ¼hrungszeit Ã¤ndern

MÃ¶chten wir jetzt zuletzt noch Ã¤ndern, wann **cron-apt** ausgefÃ¼hrt wird, mÃ¼ssen wir die entsprechende Konfigurationsdatei Ã¶ffnen. Dazu verwenden wir den folgenden Befehl:

```
sudo nano /etc/cron.d/cron-apt
```

Wir kÃ¶nnen dort jetzt die Zeit angeben wann **cron-apt** ausgefÃ¼hrt werden soll. Die Zeit geben wir Ã¼ber Syntax der **Crontabs / Cronjobs** an.

Wenn wir jetzt Ã¼berprÃ¼fen wollen, ob unser Programm sauber durchlÃ¤uft, kÃ¶nnen wir es manuell starten, mit dem folgenden Befehl.

```
sudo cron-apt -s
```

Im Weiteren legt das Programm auch Logfiles ab. Diese kÃ¶nnen wir unter `/var/log/cron-apt` einsehen.

HTTPS Reverse Proxy mit Nginx konfigurieren

Einleitung

Sobald wir **Lokal Webdienste** erstellen, mÃ¶chten wir diese vielleicht auch Ã¼ber ein **TLS Zertifikat** verschlÃ¼sseln. Dazu verwenden wir **selbst signierte Zertifikate** und einen **Nginx Webserver**. Dadurch ist es mÃ¶glich, dass wir unsere **Web-Dienste** Ã¼ber **HTTPS** erreichbar machen kÃ¶nnen.

Wie wir **TLS Zertifikate** erstellen kÃ¶nnen, wird hier genauer erklÃ¤rt.

Nginx installieren

Im ersten Schritt mÃ¼ssen wir auf unserem **Debian Server** das Paket **Nginx** installieren. Dazu verwenden wir folgenden Befehl:

```
apt update && apt upgrade -y && apt install nginx -y
```

Nginx Konfiguration anpassen

Jetzt erstellen wir die **Nginx Konfigurationsdatei** einmal neu. Im Anschluss Ã¶ffnen wir die **Konfigurationsdatei** und fÃ¼gen den nachstehenden Inhalt in die Datei ein.

```
rm /etc/nginx/sites-enabled/default
nano /etc/nginx/sites-enabled/default
```

```
server {
  listen 80;
  server_name host.name;
  return 301 https://die.domain$request_uri;
 }

server {
  listen 443 ssl;
  server_name host.name;
  ssl_certificate  /pfad/zum/zertifikat.csr;
```

```
    ssl_certificate_key  /pfad/zum/zertifikat.key;
    ssl_prefer_server_ciphers on;

    location / {
        proxy_pass http://localhost:<port>;

        proxy_set_header        Host $host;
        proxy_set_header        X-Real-IP $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Proto $scheme;
    }
}
```

Sobald wir diesen Inhalt in die Datei eingefÃ¼gt haben, brauchen wir jetzt nur noch einmal den **Nginx Dienst** neu zu starten.

```
systemctl restart nginx
```

Sobald der Dienst neu gestartet ist, Ã¼berprÃ¼fen wir, ob der Dienst ordnungsgemÃ¤ÃŸ gestartet ist. Dies kÃ¶nnen wir mit dem nachstehenden Befehl Ã¼berprÃ¼fen.

```
systemctl status nginx
```

```
* nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2022-10-27 05:47:07 UTC; 46min ago
       Docs: man:nginx(8)
    Process: 10700 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
    Process: 10702 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 10703 (nginx)
      Tasks: 2 (limit: 19031)
     Memory: 5.7M
        CPU: 244ms
     CGroup: /system.slice/nginx.service
             |-10703 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             `-10704 nginx: worker process
```

Nginx Reverse Proxy - WebSocket verliert die Verbindung

Einleitung

Konfigurieren wir unseren **Nginx Webserver** so, dass dieser als **Reverse Proxy** arbeitet, kÃ¶nnen wir bei einigen **Web-Anwendungen** das Problem haben, dass unser Client die Verbindung zum **WebSocket** verliert.

Konfiguration

Um dieses Problem zu beseitigen, mÃ¼ssen wir nur lediglich unsere **Nginx-Konfiguration** anpassen.

```
    location / {
        proxy_pass http://localhost:8080;

        proxy_http_version      1.1;
        proxy_set_header        Upgrade $http_upgrade;
        proxy_set_header        Connection "upgrade";
        proxy_set_header        Host $host;
        proxy_set_header        X-Real-IP $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Proto $scheme;
    }
```

Im Anschluss starten wir nur noch den **Nginx Webserver** neu. Danach sollte der **WebSocket** die Verbindung offen halten.

Apache2 SSL Virtualhost SSL Konfiguration

Einleitung

In diesem Beitrag befindet sich eine Version des **Apache2 Virtualhost** fÃ¼r die SSL Verbindung.

Konfigurationsdatei

```
  <VirtualHost *:80>
    ServerName FQDN
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    <Directory />
          Options FollowSymLinks
          AllowOverride None
    </Directory>
    ErrorLog  /error.log
    LogLevel warn
    CustomLog  /access.log combined
    ServerSignature On
</VirtualHost>

  <VirtualHost *:443>
    ServerName FQDN
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    <Directory />
          Options FollowSymLinks
          AllowOverride None
    </Directory>
    ErrorLog  /error.log
    LogLevel warn
    CustomLog  /access.log combined
    ServerSignature On
```

```
        SSLEngine on
        SSLCertificateFile /cert/cert.crt
        SSLCertificateKeyFile /cert/cert.key
</VirtualHost>
```