

Apache Web-Server

OS: Debian 11

Vorbereitung

1. [Basissystem einrichten](#)
2. [Firewall einrichten & aktivieren](#)
3. [SSH-Zugang einrichten](#)
 - <https://httpd.apache.org/docs/2.4/>
 - https://wiki.ubuntuusers.de/Apache_2.4/
 - <https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04>

Installation

```
sudo apt install apache2
```

Apache Status abfragen

```
sudo systemctl status apache2
```

Firewall einrichten

Firewall-Freigabe für Webseiten

```
sudo ufw allow 80/tcp
```

Firewall-Freigabe für Webseiten verschlüsselt

```
sudo ufw allow 443/tcp
```

Verzeichnis für Webseiten anlegen.

```
sudo mkdir /var/www/sites
```

Default-Einstellungen für die Webseiten-Verzeichnisse mit ACL. www-data wird als User & Gruppe vom Apache-Webbrowser genutzt. Wenn ein <FTPUSER> Daten auf den Server hochlädt, behält die Gruppe www-data und somit der Apache-Webbrowser alle erforderlichen Zugriff-Rechte um in diesem Bereich handeln zu können.

```
sudo setfacl -dm g:www-data:rwx /var/www/sites/
```

Den Main-User in die Gruppe www-data aufnehmen, damit er Schreibrechte im Bereich der HTML-Seiten hat.

```
sudo usermod -aG www-data <SUDO-USER>
```

FTP-Zugang

```
sudo apt install vsftpd
```

```
sudo nano /etc/vsftpd.conf
```

```
local_enable=YES  
anonymus_enable=NO  
write_enable=YES
```

```
sudo systemctl reload vsftpd
```

FTP-User anlegen - Namen festlegen. Der Zugriff für den <FTPUSER> wird auf den u.a. Bereich eingeschränkt, das gilt auch für den SSH-Zugriff über die Shell. Der o.a. <SUDO-USER> sollte daher nicht hier verwendet werden.

```
sudo adduser <FTPUSER> --no-create-home
```

FTP-User in die Apache-Gruppe aufnehmen

```
sudo usermod -aG www-data <FTPUSER>
```

Zugang für FTP-User einschränken auf den Bereich der Webseiten

```
sudo nano /etc/ssh/sshd_config
```

```
Match User <FTPUSER>  
  X11Forwarding no  
  AllowTcpForwarding no  
  PermitTTY no  
  ForceCommand internal-sftp  
  ChrootDirectory /var/www/  
  PasswordAuthentication yes
```

```
sudo systemctl reload ssh
```

Für den FTP-Zugriff von <FTPUSER> mit o.a. Einschränkung (Match User ...) ist eine Besonderheit zu beachten, da sonst kein Zugang per FTP möglich ist.

Der Owner vom Verzeichnis /var/www, bzw. Pfad aus Parameter **ChrootDirectory**, muss root sein und Gruppe oder Sonstige dürfen keine Schreibrechte besitzen.

Diese Vorgaben gelten nur für das Hauptverzeichnis, nicht für darin enthaltene Unterverzeichnisse, daher im folgenden die Option -R nicht angewandt.

Falls das nicht gegeben ist:

Setze Besitzer.

```
sudo chown root:root /var/www/
```

Entferne Schreibrechte (w) für Gruppe (g)

```
sudo chmod g-w /var/www/
```

Entferne Schreibrechte für Sonstige (o)

```
sudo chmod o-w /var/www/
```

Anschließend hat der FTP-User Lese-Zugriff auf das Verzeichnis `/var/www`. Schreibrechte müssen dann in weiteren Unterverzeichnissen (`/var/www/html`, `/var/www/sites`, etc.) für die Gruppe `www-data` erteilt werden bzw. müssten eigentlich bereits für den Apache-Webserver vorhanden sein - siehe auch oben unter [Installation](#).

Webseiten & virtueller Host

Verzeichnis für die HTML-Daten über die Shell anlegen

```
sudo mkdir /var/www/sites/KSPI.DE
```

```
sudo chmod -R 775 /var/www/sites/KSPI.DE
```

Die Programm-Dateien können dann mit einem FTP-Programm übertragen werden.

Je nach Anwendungsfall ist es sinnvoll sensible Daten außerhalb des direkten Webseitenzugriffs und in andere Verzeichnisse zu platzieren. Der Apache-Webbrowser (User `www-data`) muss Zugriff darauf haben → Z.B. im Verzeichnis `/var/www/data/...`. Das muss über die Konfiguration der jeweilige Webseite definiert werden. Z.B. bei Nextcloud über die `...config/config.php`

```
sudo mkdir /var/www/data/
```

Auch hier kann man wie bereits im Verzeichniss `/var/www/sites/` den Bereich für die Gruppe `www-data` per Default mit Rechten versehen.

```
sudo setfacl -dm g:www-data:rwx /var/www/data/
```

Alternativ:

```
sudo chown -R www-data:www-data /var/www/data/
```

und

```
sudo chmod -R 775 /var/www/data/
```

Apache-Konfiguration virtueller Host der Domain

```
sudo nano /etc/apache2/sites-available/KSPI.DE.conf
```

```
<VirtualHost *:80>
  ServerName KSPI.DE
  ServerAdmin mail@KSPI.DE
  DocumentRoot /var/www/sites/KSPI.DE
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined
  <Directory /var/www/sites/KSPI.DE>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Order allow,deny
    allow from all
  </Directory>
</VirtualHost>
```

Beim Erstellen der Let's-Encrypt-Zertifikate gab es Probleme mit dem ServerAlias „www....“. Daher wurde hier auf die Einstellung des ServerAlias verzichtet. Es ist möglich, die www.<DOMAIN> als eigenständige Subdomain zu verarbeiten (s.u.) und dafür eine eigenes Zertifikat zu erstellen. Für Documentroot wird das selbe Verzeichnis angegeben.

Virtuellen Host im Apache-Web-Server registrieren

```
sudo a2ensite KSPI.DE.conf
```

Defaultseite deaktivieren. Dadurch wird erste Domain (nach Alphabet) zur Defaultseite, sollte z.B. nur die Server-IP-Adresse über einen Webbrowser aufgerufen werden, oder die per DNS hierhin umgeleitete Domain nicht zu finden sein.

```
sudo a2dissite 000-default.conf
```

Alternativ kann eine permanente Umleitung in diese Datei geschrieben werden, um eine bestimmte Domain aufzurufen.

```
<VirtualHost *:80>
  RedirectPermanent / https://duckduckgo.de/
</VirtualHost>
```

Hostname und Fully-Qualified Host Name (FQHN) festlegen

```
sudo hostnamectl set-hostname server.KSPI.DE
```

Einstellungen für den Apacheserver testen

```
sudo apache2ctl configtest
```

Sollte es Probleme mit dem FQDM¹⁾ geben, dann ggf. die Datei hosts anpassen. Reihenfolge beachten.

```
sudo nano /etc/hosts
```

```
127.0.1.1 server.KSPI.DE server
```

```
127.0.0.1 localhost
```

Wenn Test ok, dann alles aktivieren

```
sudo systemctl reload apache2
```

Bei Apache registrierte Hosts

```
sudo apache2ctl -S
```

Bei Apache registrierte Module

```
sudo apache2ctl -M
```

Mit FTP-Zugang die Struktur anlegen und Dateien übertragen

Simple HTML-Datei

```
sudo nano /var/www/sites/KSPI.DE/index.html
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>TEST-TITEL</title>
  </head>
  <body>
    <center>
      <h1>TESTSEITE auf Apache-Server erreichbar</h2>
    </center>
  </body>
</html>
```

SSL-Zertifikate

<https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu-20-04-de>

DNS-Einstellungen

Die DNS-Server der Domain(s) müssen vorher beim Domainprovider eingestellt sein:

- A → @ → IPV4-Adresse des Servers
- A → www → IPV4-Adresse des Servers (sofern www gewünscht)

Werden Subdomains eingerichtet zusätzlich:

- A → * → IPV4-Adresse des Servers

Let's-Encrypt

```
sudo apt install certbot python3-certbot-apache
```

Zertifikat(e) erstellen

```
sudo certbot --apache
```

Durch das Erstellen eines Zertifikats wird automatisch zusätzlich eine Datei für den virtuellen Host der SSL-Verbindung angelegt. Z.B. für die Hostdatei „KSPI.DE.conf“, zusätzlich hinzu „/etc/apache2/sites-available/KSPI.DE-le-ssl.conf“²⁾.

Timer-Einstellungen für Autorenew

```
sudo systemctl status certbot.timer
```

Erneuerung Testen

```
sudo certbot renew --dry-run
```

Zertifikate anzeigen

```
sudo certbot certificates
```

Zertifikat löschen

```
sudo certbot delete
```

Weitere Domains

Hier als Beispiel die Subdomain für das wiki > <https://wiki.KSPI.DE>

Verzeichnis für die Daten anlegen

```
sudo mkdir /var/www/sites/wiki.KSPI.DE
```

Die Programm-Dateien können dann mit einem FTP-Programm übertragen werden.

Apache-Konfiguration virtueller Host der Domain

```
sudo nano /etc/apache2/sites-available/wiki.KSPI.DE.conf
```

```
<VirtualHost *:80>
  ServerName wiki.KSPI.DE
  ServerAdmin mail@KSPI.DE
  DocumentRoot /var/www/sites/wiki.KSPI.DE
```

```
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
<Directory /var/www/sites/wiki.KSPI.DE>
  Options Indexes FollowSymLinks MultiViews
  AllowOverride All
  Order allow,deny
  allow from all
</Directory>
</VirtualHost>
```

Virtuellen Host im Apache-Web-Server registrieren

```
sudo a2ensite wiki.KSPI.DE.conf
```

Einstellungen für den Apacheserver testen

```
sudo apache2ctl configtest
```

Wenn Test ok, dann alles aktivieren

```
sudo systemctl reload apache2
```

SSL-Zertifikat registrieren

```
sudo certbot --apache
```

PHP

Ubuntu

<https://php.watch/articles/install-php82-ubuntu-debian>

Debian

Der Apache-Server kann parallel mit unterschiedlichen PHP-Versionen betrieben und eine Version davon einer Domain explizit zugewiesen werden.

Quelle:

<https://www.codinghood.de/blog/2021/05/php-7-4-und-php-8-0-zeitgleich-auf-einem-debian-10-mit-apache-und-php-fpm-installieren/>

Benötigte Pakete wie curl installieren

```
sudo apt-get -y install apt-transport-https lsb-release ca-certificates curl
```

Mit dem sury PHP Repository kommunizieren > Schlüssel herunterladen und installieren

```
sudo curl -sSL -o /etc/apt/trusted.gpg.d/php.gpg
```

```
https://packages.sury.org/php/apt.gpg
```

sury PHP Repository zur Source-List hinzufügen

```
sudo sh -c 'echo "deb https://packages.sury.org/php/ $(lsb_release -sc)
main" > /etc/apt/sources.list.d/php.list'
```

Paketlisten neu einlesen und aktualisieren.

```
sudo apt-get update
```

PHP-Versionen installieren (je nach Bedarf)

```
sudo apt install php7.4 php7.4-bcmath php7.4-cli php7.4-common php7.4-curl
php7.4-dev php7.4-fpm php7.4-gd php7.4-imagick php7.4-intl php7.4-json php7.4-
mbstring php7.4-mysql php7.4-opcache php7.4-readline php7.4-soap php7.4-
sqlite3 php7.4-tidy php7.4-xsl php7.4-zip
```

```
sudo apt install php8.0 php8.0-bcmath php8.0-cli php8.0-common php8.0-curl
php8.0-dev php8.0-fpm php8.0-gd php8.0-imagick php8.0-intl php8.0-mbstring
php8.0-mysql php8.0-opcache php8.0-readline php8.0-soap php8.0-sqlite3
php8.0-tidy php8.0-xml php8.0-xsl php8.0-zip
```

```
sudo apt install php8.2 php8.2-bcmath php8.2-cli php8.2-common php8.2-curl
php8.2-dev php8.2-fpm php8.2-gd php8.2-imagick php8.2-intl php8.2-mbstring
php8.2-mysql php8.2-opcache php8.2-readline php8.2-soap php8.2-sqlite3
php8.2-tidy php8.2-xml php8.2-xsl php8.2-zip
```

Check Installiert - sollte bei allen Versionen klappen

```
sudo systemctl status php7.4-fpm.service
```

Apache-Module zum Umgang mit den PHP-FPM Installationen

```
sudo apt-get install libapache2-mod-fcgid php-fpm
```

```
sudo a2enmod actions alias fcgid proxy_fcgi
```

Version auswählen (Server intern)

```
sudo update-alternatives --config php
```

Auswahl Basisversion (Server intern)

```
sudo update-alternatives --config php-fpm.sock
```

Standartversion Apache festlegen → hier nur eine Version, die anderen, sofern aktiviert, mit a2disconf abschalten


```
sudo a2enconf php8.2-fpm.conf
```

```
sudo systemctl reload apache2
```

Nur wenn eine Domain/Subdomain eine andere, als die PHP-Standard-Version, benötigt, die entsprechende Apache-Conf-Datei konfigurieren und Zeile einfügen.

```
sudo nano /etc/apache2/sites-available/xxx.conf
```

Zum Beispiel:

```
Include /etc/apache2/conf-available/php8.0-fpm.conf
```

```
sudo systemctl reload apache2
```

Check PHP-Version auf Webseite

```
sudo nano /var/www/sites/.../phpinfo.php
```

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

MySQL-Datenbank

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

```
sudo mysql_secure_installation
```

„Altes“ root-Passwort ist leer, alle Fragen mit Y beantworten und neues Passwort eintragen.

Standardverzeichnis für Datenbanken

```
/var/lib/mysql
```

PHPMyAdmin

Es ist möglich, die PHPMyAdmin von Debian zu verwenden > `apt install phpmyadmin`. Alternativ (hier angewandt) die aktuelle Variante von der Internetseite von `phpmyadmin.net` verwenden. Aktuelle Installationsdateien downloaden in Unterverzeichnis des Apacheservers, oder holen mit „wget“. Pfad und Dateiname siehe <https://www.phpmyadmin.net/>

```
cd /var/www/sites/phpmyadmin/
```

In diesem Beispiel wird eine bestehende Domain genutzt und darin das Unterverzeichnis `phpmyadmin` erstellt. Aufruf über den Browser mit `<DOMAIN>/phpmyadmin`.

Alternative: Dateien in das Hauptverzeichnis einer Subdomain legen. Aufruf mit `<SUB>.<DOMAIN>`

(ohne Erweiterung).

Download der Installations-Dateien

```
sudo wget
https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-all-languages
.zip
```

Sollte unzip nicht installiert sein:

```
sudo apt-get install unzip
```

```
sudo unzip phpMyAdmin-5.2.1-all-languages.zip
```

```
sudo mv phpMyAdmin-5.2.1-all-languages/ phpmyadmin/
```

```
sudo chown -R www-data:www-data /var/www/sites/phpmyadmin
```

```
sudo chmod -R 775 /var/www/sites/phpmyadmin
```

```
cd phpmyadmin/
```

```
sudo cp config.sample.inc.php config.inc.php
```

```
sudo nano config.inc.php
```

ÄNDERN IN:

```
$cfg['blowfish_secret'] =
sodium_hex2bin('f16ce59f45714194371b48fe362072dc3b019da7861558cd4ad29e4d6fb1
3851');
```

Sicherheit: direkten root-Zugang über phpmyadmin sperren.

Möglich über phpmyadmin oder wie folgt:

```
sudo nano /var/www/sites/phpmyadmin/config.inc.php
```

Entscheidung über letzte Zeile (AllowRoot = false == gesperrt) ggf. Zeile einfügen:

```
/* Configure according to dbconfig-common if enabled */
if (!empty($dbname)) {
    /* Authentication type */
    $cfg['Servers'][$i]['auth_type'] = 'cookie';
    $cfg['Servers'][$i]['AllowRoot'] = false;
    ...
}
```

Neuen User anlegen:

```
sudo mysql
```

```
CREATE USER '<USERNAME>'@'localhost' IDENTIFIED BY '<PASSWORD>';
```

User mit Rechten versehen (hier alle):

```
GRANT ALL PRIVILEGES ON * . * TO '<USERNAME>'@'localhost' WITH GRANT OPTION;
```

Berechtigungen neu laden:

```
FLUSH PRIVILEGES;
```

1)

Fully qualified domain name

2)

le für Let's-Encrypt

From:

<https://wiki.bluegnu.de/> - **gniki**

Permanent link:

<https://wiki.bluegnu.de/doku.php?id=open:it:apache&rev=1720890561>

Last update: **2024/07/13 19:09**

