Apache Web-Server

Installation unter Debian 11

Apache Web Server

Siehe https://httpd.apache.org/docs/2.4/
oder https://wiki.ubuntuusers.de/Apache_2.4/

Apache Webserver installieren[1)]

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install apache2
$ sudo a2enmod rewrite
$ sudo a2enmod actions
$ sudo a2enmod ssl
$ sudo systemctl restart apache2
```

Zugriffsrechte mit ACL

Damit unterschiedliche Nutzer per SFTP Dateien hochladen können und diese für alle Beteiligten editierbar bleiben, werden die Verzeichnisrechte für den Web-Bereich mit ACL verwaltet. Für den externen Zugriff siehe SSH einrichten.
Mit Access Control Lists (ACL) ist es möglich, einzelnen Nutzern (oder auch Gruppen) gezielt Rechte an einzelnen Dateien zu geben oder zu entziehen. Durch die folgende Einrichtung wird der Gruppe www-data das Lese- und Schreibrecht auf das Verzeichnis /var/www/ und seiner Unterverzeichnisse gewährt. Durch die ACL erhalten das auch alle neu erstellen Dateien und Ordner.
Die Gruppe www-data ist die Linux-Gruppe für den Apache-Server. User, die Zugriff per SFTP auf die Bereiche benötigen, müssen als User im System angelegt werden.

Gruppe www-data zum Eigentümer der Web-Verzeichnisse erklären

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

Lese und Schreib-Rechte für die Gruppe auf die aktuellen Verzeichnisse, Unterverzeichnisse und Dateien setzen

```
$ sudo chmod -R g+rw /var/www/
```

ACL installieren

```
$ sudo apt-get install acl
```

Defaultvorgaben für die Gruppe www-data setzen im Verzeichnis /var/www

```
$ sudo setfacl –dm g:www-data:rw /var/www/
```

-d = Default-Vorgaben (auch für Unterverzeichnisse)
-m = Maske

User in die Gruppe www-data aufnehmen

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

PHP

```
$ sudo apt-get install ca-certificates apt-transport-https software-
properties-common -y
$ sudo echo "deb https://packages.sury.org/php/ $(lsb_release -sc) main" |
tee /etc/apt/sources.list.d/sury-php.list
$ sudo wget -qO - https://packages.sury.org/php/apt.gpg | apt-key add -
$ sudo apt-get update -y
$ sudo apt-get install php8.0
$ php -v
```

php8.0 ggf. ersetzen mit gewünschter Version
php -v → Check aktuelle PHP-Version

Datenbank

MariaDB ist ein freies, relationales Open-Source-Datenbankmanagementsystem, das durch eine Abspaltung aus MySQL entstanden ist.
Siehe: https://mariadb.com/

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

Passwort bei Installation ist leer (Enter), alles andere mit Y bestätigen und neues PW vergeben.

Datenbankverwaltung

phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the Web.
Siehe: https://www.phpmyadmin.net/

```
$ sudo apt-get install phpmyadmin
$ sudo systemctl reload apache2
```

Sicherheit: direkten root-Zugang über phpmyadmin sperren - ggf vorher einen anderen User zum Admin machen.

Möglich über phpmyadmin oder wie folgt:

```
$ sudo nano /etc/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;
      ...
```

Virtueller Host

Beim Provider der Domain wird per DNS-Eintrag auf die IP-Adresse des Servers verwiesen. Die virtuellen Hosts werten bei eingehenden Anfragen den Domainnamen aus und leiten in das jeweils zugeordnete Verzeichnis.

Registrierung der Domainsnamen im Verzeichnis /etc/apache2/sites-available/
Zur Bearbeitung die Vorgabedateien (default) kopieren und mit eigenen Einstellungen anpassen.
Um einen guten Überblick zu halten, ist es sinnvoll, für jeden Host eine eigener Datei zu erstellen:
Endung auf .conf - Gute Identifizierung besteht, wenn Dateiname dem Domainnamen entspricht.

Bei namensbasierten Systemen wird für alle Port 80 (Standard) und Port 443 (SSL) angewandt.
Alternativ ist die Unterscheidung über verschiedene Ports möglich. Die Ports müssen dann in der Datei /etc/apache2/ports.conf hinterlegt sein.
Hier betrachtetes System ist Namensbasiert.

Beim Eingang einer Anforderung muss der Domainname mitgeliefert werden (Provideraufgabe), so dass der virtuelle Host die Anfrage verarbeiten kann.

SSL-Beispiel:

```
$ sudo nano /etc/apache2/sites-available/EXAMPLE.com.conf
```

```
<VirtualHost *:80>
  #Web-Adresse (http:// ohne SSL) die hier verarbeitet wird
  ServerName EXAMPLE.com
  #auch die www-Adresse wird hier verarbeitet
  ServerAlias www.EXAMPLE.com
  #Permanente Umleitung auf die SSL-Adresse
  RedirectPermanent / https://EXAMPLE.com/
</VirtualHost>
<IfModule mod_ssl.c>
  <VirtualHost _default_:443>
    #Web-Adresse (https:// mit SSL) die hier verarbeitet wird
    ServerName EXAMPLE.com
    #Auch die www-Adresse wird hier verarbeitet
```

```
    ServerAlias www.EXAMPLE.com
    #Verzeichnis, welches den Haupt-Dokumentenbaum bildet, der im Web
sichtbar ist.
    #Dort sollte die index.html bzw. index.php liegen
    #z.B. zeigt der Pfad bei Shopware auf das Unterverzeichnis public
    DocumentRoot /var/www/PERSONAL/EXAMPLE.com/public
    #E-Mail-Adresse, die der Server in Fehlermeldungen einfügt, welche an
den Client gesendet werden
    ServerAdmin admin@EXAMPLE.com
    #Ablageort, an dem der Server Fehler protokolliert
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
    #Steuert die Ausführlichkeit des Fehlerprotokolls. Debug-Level-
Nachrichten = tiefstes Level (alles)
    LogLevel debug
    #SSL-Verwaltung & Ablageort der Zertifikate. Hier wird "let's encrypt"
genutzt
    SSLEngine on
    SSLCertificateFile
/opt/letsencrypt/cert/live/EXAMPLE.com-0001/fullchain.pem
    SSLCertificateKeyFile
/opt/letsencrypt/cert/live/EXAMPLE.com-0001/privkey.pem
    #<Directory> Umschließt eine Gruppe von Direktiven, die nur auf das
genannte Verzeichnis des Dateisystems und Unterverzeichnisse angewendet
werden
    <Directory /var/www/PERSONAL/EXAMPLE.com>
      #Options: Erlaubt die Verwendung von Direktiven zur Steuerung
spezieller Verzeichniseigenschaften
      #Indexes: Erlaubt die Verwendung von Direktiven zur Steuerung von
Verzeichnisindizes
      #FollowSymLinks: Der Server folgt symbolischen Links in diesem
Verzeichnis.
      Options Indexes FollowSymLinks MultiViews
      #Wenn diese Anweisung auf All gesetzt wird, dann ist jede Direktive in
den .htaccess-Dateien erlaubt, die den Kontext .htaccess besitzt.
Alternative = "None".
      AllowOverride All
      Order allow,deny
      allow from all
    </Directory>
  </VirtualHost>
</IfModule>
```

Seiten beim Apache registrieren[2)].
REGISTRIEREN:

```
$ sudo a2ensite EXAMPLE.com.conf
```

AUSTRAGEN:

```
$ sudo a2dissite EXAMPLE.com.conf
```

Einstellungen testen:

```
$ sudo apache2ctl configtest
```

Apache neu starten:

```
$ sudo systemctl reload apache2
```

SSL-Zertifikat

Die Zertifikate müssen entweder extern beschafft oder über das System registriert werden (z.B mit let's encrypt) .
Externe Zertifikate und Privat-Keys werden gespeichert (z.B. im Verzeichnis /etc/apache2/ssl/) und in der Datei domain.conf des virtuellen Hosts von Apache (s.o.) mit Pfadangabe registriert. Zertifikat mit verschiedenen Endungen: crt, cer, etc, funktionieren. Die Zertifikate müssen ggf. regelmäßig erneuert werden.

Let's Encrypt

Folgende Beschreibung geht davon aus, dass die Domain bei Ionos registriert ist. Für andere Provider gibt es andere Wege oder ggf. eigene Plugins - wie z.B. bei cloudflare.

- https://certbot.eff.org/
- https://eff-certbot.readthedocs.io/en/stable/using.html#dns-plugins
- Quelle: https://www.devlix.de/lets-encrypt-wildcard-zertifikate-mit-ionos-dns-api-erzeugen/

Voraussetzung hier, dass die Domains bei Ionos registriert sind und der API-Zugriffsschlüssel von Ionos vorhanden ist. Der API-Zugriffsschlüssel ist eine (derzeit kostenlose) Zusatzleistung bei Ionos und muss ggf. erst aktiviert werden - ggf. telefonisch über den Support.

Der Einfachheit halber nutzen wir das offizielle Certbot Docker Image. Dort ist bereits das Tool certbot und python installiert. Damit das Zertifikat erstellt werden kann, müssen verschiedene Dinge vorbereitet werden.

- Ein Ordner für die Let’s Encrypt Zertifikat Struktur muss erstellt werden. In unserem Beispiel liegt dieser unter /opt/letsencrypt/cert
- Ein Ordner, in dem die Skripte abliegen. In unserem Beispiel liegen diese unter /opt/letsencrypt/scripts und sind ausführbar (chmod +x /opt/letsencrypt/scripts/*.sh)
    - /opt/letsencrypt/scripts/authenticate.sh
    - /opt/letsencrypt/scripts/cleanup.sh
- Einerseits muss der API Zugriff beantragt worden sein (siehe oben) und andererseits muss der <publicprefix> und das <secret> im folgenden Codeblock entsprechend ersetzt werden.
    - HINWEIS: Es muss ein Punkt (.) zwischen publicprefix und secret gesetzt werden!
- Eine Email für Expiration Benachrichtigungen und als Identifier für einen Let’s Encrypt Account muss im folgenden Codeblock (<email-address>) entsprechend ersetzt werden
- Die Domain, für die das Zertifikat erstellt werden soll, muss im folgenden Codeblock ersetzt

werden (<your.domain>). Erster Teil für die TLD[3], zweiter Teil (expand) für alle Sub-Domains (inkl. www.)

```
$ sudo apt-get install Docker
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Für jede Domain, jeden virtuellen Host dieses Servers, (kann) eine eigene Datei (<your.domain>.sh) erstellt werden. Die Dateien können dann direkt aufgerufen werden:

```
$ sudo bash ./opt/letsencrypt/scripts/<your.domain>.sh
```

Damit das Ablaufen des Zertifiktes verhindert wird, kann dies automatisiert werden über einen Cronjob

```
$ sudo nano /etc/crontab
```

```
# /etc/crontab: system-wide crontab
0 5 2 * *   root    /opt/letsencrypt/scripts/<your.domain>.sh
```

Ausführung:

- Jeden 2. Tag im Monat um 5:00 Uhr

```
$ sudo nano /opt/letsencrypt/scripts/<your.domain>.sh
```

```
docker run -i --rm \
-v /opt/letsencrypt/cert:/etc/letsencrypt \
-v /opt/letsencrypt/scripts:/tmp/scripts \
-e "API_KEY=<publicprefix>.<secret>" \
certbot/certbot \
certonly \
--keep-until-expiring \
--preferred-challenges dns \
--non-interactive \
--agree-tos \
-m <email-address> \
--manual \
--manual-auth-hook /tmp/scripts/authenticate.sh \
--manual-cleanup-hook /tmp/scripts/cleanup.sh \
-d <your.domain> --expand -d *.<your.domain>
```

Ersetzen: <publicprefix>, <secret>, <email-address>, 2x <your.domain>

Über o.a. <your.domain>.sh werden zwei weitere Programme aufgerufen.
Der Inhalt kann (aktuell) ohne Anpassungen übernommen werden.
Diese drei Dateien/Progamme müssen ausführbar sein !!

```
$ sudo nano /opt/letsencrypt/scripts/authenticate.sh
```

```
#!/bin/sh
```

```
#set -x
apk add curl
API_URL="https://api.hosting.ionos.com/dns/v1"
API_KEY_HEADER="X-API-Key: $API_KEY"

# Strip only the top domain to get the zone id
ZONE_NAME=$(expr match "$CERTBOT_DOMAIN" '.*\.\(.*\..*\)')
# When already the TLD then use it
if [ -z "$ZONE_NAME" ]; then
  ZONE_NAME="$CERTBOT_DOMAIN"
fi

# Get the Ionos zone id
ZONE_RESPONSE=$(curl -s -X GET "$API_URL/zones" \
                  -H "$API_KEY_HEADER" \
                  -H "Accept: application/json")
ZONE_ID=$(echo $ZONE_RESPONSE \
   | python -c "import sys,json;response=json.load(sys.stdin);print(next((x
for x in response if x['name']=='$ZONE_NAME'))['id'])")

# Create TXT record
CREATE_DOMAIN="_acme-challenge.$CERTBOT_DOMAIN"
RECORD_CREATE_RESPONSE=$(curl -s -X POST "$API_URL/zones/$ZONE_ID/records" \
                           -H "$API_KEY_HEADER" \
                           -H "Content-Type: application/json" \
                           --data '[{"name": "'"$CREATE_DOMAIN"'", "type":
"TXT", "content": "'"$CERTBOT_VALIDATION"'", "ttl": 3600, "prio": 100,
"disabled": false}]')

# Save info for cleanup
echo $ZONE_ID > /tmp/CERTBOT_$CERTBOT_DOMAIN

# Sleep to make sure the change has time to propagate over to DNS
sleep 25
```

```
$ sudo nano /opt/letsencrypt/scripts/cleanup.sh
```

```
#!/bin/sh
#set -x

API_URL="https://api.hosting.ionos.com/dns/v1"
API_KEY_HEADER="X-API-Key: $API_KEY"

if [ -f /tmp/CERTBOT_$CERTBOT_DOMAIN ]; then
  ZONE_ID=$(cat /tmp/CERTBOT_$CERTBOT_DOMAIN)
  rm -f /tmp/CERTBOT_$CERTBOT_DOMAIN

  CREATE_DOMAIN="_acme-challenge.$CERTBOT_DOMAIN"
  # request the created records
  RECORD_GET_RESPONSE=$(curl -s -X GET
"$API_URL/zones/$ZONE_ID?recordName=$CREATE_DOMAIN&recordType=TXT" \
```

```
                          -H "$API_KEY_HEADER" \
                          -H "Accept: application/json")
  RECORD_IDS=$(echo $RECORD_GET_RESPONSE \
          | python -c "import
sys,json;records=json.load(sys.stdin)['records'];print('\n'.join([x['id']
for x in records]))")
fi

# Remove the challenge TXT record from the zone
if [ -n "$ZONE_ID" -a -n "$RECORD_IDS" ]; then
  echo "$RECORD_IDS" \
  | xargs -n1 -I {} curl -s -X DELETE "$API_URL/zones/$ZONE_ID/records/{}" \
          -H "$API_KEY_HEADER"
fi
```

Backup

Auf externem Server

a) Die Dateien der Webseiten (Verzeichnis /var/www/) und die Datenbanken von MariaDB.

Automatisiert ausführen über Cronjob

```
$ sudo nano /etc/crontab
```

```
# /etc/crontab: system-wide crontab
0 4 * * *   root    /PFAD/backup_dbase.sh
0 3 1 * *   root    /PFAD/backup_www.sh
```

Ausführung:

- jeden Tag um 4:00 Uhr die Datenbank
- jeden 1. Tag im Monat um 3:00 Uhr die Dateien aus Verzeichnis www

Alle *.sh-Dateien müssen ausführbar sein!

```
$ sudo nano /PFAD/backup_dbase.sh
```

```
#!/bin/bash
#Legt bis zu 31 Sicherungen an, jeweils mit Tagesdatum.
#Keine Angabe von Monat oder Jahr.
#Bei demselben Datum wird überschrieben
BACKUP_DIR="/PFAD/backup"
DAY=$(date +%d)
cd /
#Verzeichnis anlegen, falls nicht existent
mkdir -p ${BACKUP_DIR}
#Komplette Datenbank in SQL-File sichern
mysqldump -uUSERNAME -pPASSWORT ${DATENBANK} > ${BACKUP_DIR}/${DATENBANK}-
```

```
${DAY}.sql
exit
```

Ersetzen: PFAD, DATENBANK, USERNAME (-u davor), PASSWORT (-p davor).
Hier Beispielhaft für eine Datenbank. Vorletzte Zeile „mysqldump .. .“ einfach vervielfätligen, um weitere Datenbanken zu sichern.

```
$ sudo nano /PFAD/backup_www.sh
```

```
#!/bin/bash
SOURCE="/var/www/ "
BACKUP_DIR="/PFAD/backup"
cd /
#Verzeichnis anlegen, falls nicht existent
mkdir -p ${BACKUP_DIR}
#Daten packen und in einer Datei sichern
tar -cpzf ${BACKUP_DIR}/backup_data.tar.gz ${SOURCE}
exit
```

Ersetzen: PFAD

b) Die System-Dateien (Hier eine kleine Auswahl).
Regelmäßig (nach Änderungen) mit root-Rechten ausführen.

```
$ sudo bash ./PFAD/backup_system.sh
```

```
$ sudo nano /PFAD/backup_system.sh
```

```
#!/bin/bash
#   -a / --archiv           Übernimmt Rechte und Metadaten
#   -p / --preserve         Attribute der Originaldatei werden beim Kopieren
vererbt
#   --preserve=timestamp    Zeitstempel der Originaldatei wird beim Kopieren
vererbt
#   -r / -R / --recurcive   Verzeichnisse inkl. Unterverzeichnisse
#   -u / --update           Kopiert die Datei nur, wenn die Zieldatei älter
als das Original ist
#   -v  / --verbose         Gibt nach dem Kopiervorgang eine Meldung aus
# Werden (nur) einzelne Dateien kopiert, müssen die Zielverzeichnisse
bereits existieren
cp -a -p -u -v /etc/crontab  /PFAD/backup/system/
cp -a -p -u -v /etc/ssh/sshd_config  /PFAD/backup/system/
cp -r -a -u -p -v /etc/apache2/sites-available/  /PFAD/backup/system/
cp -r -a -u -p -v /opt/letsencrypt/  /PFAD/backup/system/
exit
```

Ersetzen: PFAD

Auf lokalen Linux-PC

Von einem lokalen Debian-/Linux-System kann das Backup kopiert werden. Der Login muss mit einer SSH-Zertifikatsdatei ohne Passwort für den im Cronjob stehenden User realisiert sein.

Automatisiert über Cronjob

```
$ sudo nano /etc/crontab
```

```
# /etc/crontab: system-wide crontab
0 4 * * 7               USERNAME      /PFAD/backup_externer_server.sh
```

Ausführung:

- jeden 7. Tag der Woche um 4:00 Uhr

```
$ sudo nano /PFAD/backup_externer_server.sh
```

```
#!/bin/bash
scp -r USERNAME@IPADRESSE:/PFADEXTERN/backup/ /PFADLOKAL/
exit
```

Ersetzen: PFADEXTERN, PFADLOKAL, USERNAME, IPADRESSE < externer Server
scp = Kopieren über gesicherte ssh-Verbindung
-r = Kopieren inkl. aller Unterverzeichnisse

Achtung: Es kann nur kopiert werden, wenn auch Zugriff von USERNAME auf die externen Dateien besteht. Sollen z.B. die letsencrypt-Zertifikate kopiert werden, dann müssen hierfür am externen System Zugriffsrechte vorliegen.

Begriffe

Linux

APT: Advanced Packaging Tool/ Packetverwaltung.
apt-get: Befehle der Packetverwaltung (installieren, update, etc.)
sudo: Ausführen mit root-Rechten.
nano: einfacher Texteditor.
systemctl ist ein Kommandozeilenwerkzeug für systemd. Mit Hilfe von systemctl können Befehle an systemd gesendet werden, z.B. zum Steuern von Units, zum Abfragen des Status, zum Herunterfahren des Systems etc.

Apache Web-Server

a2enmod, a2dismod, a2ensite oder a2dissite
a2: apache 2
en: enable/aktivieren

dis: disable/deaktivieren
mod: Modus
site: Seite [4].
rewrite: Modul, um URLs zu manipulieren. Mit der RewriteEngine des Apache 2.4 Webservers ist es möglich, die angeforderte URL anhand von Regeln „umzuschreiben“ (auf englisch: to rewrite).
Siehe https://wiki.ubuntuusers.de/Apache/mod_rewrite/
actions: Modul ermöglicht die Ausführung von CGI-Skripten in Abhängigkeit von Medientypen und Anfragemethoden.
ssl: SSL Verbindungen auf Apache aktivieren

[1]
Siehe #Begriffe
[2], [4]
muss als *.conf-Datei im Pfad: /etc/apache2/sites-available/ zur Verfügung stehen
[3]
Top Level Domain