

Apache2-Webserver & Benutzerverwaltung im LXC-Container

Diese Anleitung zeigt dir Schritt für Schritt, wie du in einem bereits eingerichteten Ubuntu-LXC-Container ([apache110](#)) den Apache2-Webserver installierst, Web-Content erstellst, PHP installierst und gebrauchst, Benutzer mit verschiedenen Rechten erstellst und Apache-Aliase konfigurierst.

Bevor wir beginnen loggen wir uns über das Webinterface auf den Proxmox Server und gehen über Proxmox auf die Console vom Container 110. Jetzt loggen wir uns als pdal mit dem vorher festgelegtem Passwort ein.

```
apache110 login: pdal
Password:
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.12-11-pve x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

1. Apache2 installieren

Einführung zu Apache2

Apache2 ist ein weit verbreiteter **Webserver**, der HTTP-Anfragen von Clients (z.B. Webbrowsern) entgegennimmt und darauf basierend Inhalte ausliefert. Er gehört zur **Open-Source-Software-Familie** und wird auf vielen Betriebssystemen, insbesondere Linux, eingesetzt.

Hauptfunktionen von Apache2

- **Bereitstellung von Webseiten:** Stellt HTML-, PHP- oder andere Webinhalte über HTTP/HTTPS bereit.
- **Verwaltung von Webanwendungen:** Ermöglicht das Hosten dynamischer Webanwendungen, oft in Verbindung mit PHP, Python oder anderen serverseitigen Sprachen.
- **Virtuelle Hosts:** Unterstützt mehrere Webseiten auf einem Server, jede mit eigenem Domainnamen und Webverzeichnis.
- **Sicherheit und Konfiguration:** Bietet Module für Authentifizierung, Zugriffskontrolle, Verschlüsselung (SSL/TLS) und Logging.

Apache2 wird eingesetzt, um **Webseiten und Webanwendungen für Benutzer zugänglich** zu machen. Er verarbeitet Anfragen, liefert die entsprechenden Dateien aus und kann Inhalte dynamisch generieren, z.B. durch Integration mit PHP.

Kurzer Vergleich zu Nginx

- **Architektur:** Apache2 verwendet in der Standardkonfiguration einen **prozessbasierten oder Thread-basierten Ansatz**, während Nginx ein **ereignisgesteuertes, asynchrones Modell** nutzt.
- **Leistung:** Nginx ist bei hohem Traffic oft **ressourcenschonender und schneller** bei statischen Inhalten, Apache2 bietet mehr Flexibilität bei dynamischen Inhalten.
- **Konfiguration:** Apache2 setzt stark auf modulare Konfiguration pro Verzeichnis (.htaccess), Nginx verwendet zentralisierte Konfigurationsdateien.
- **Einsatzbereiche:** Apache2 eignet sich gut für klassische PHP-Webanwendungen, Nginx oft als **Reverse Proxy** oder für stark skalierte Webdienste.

Paketlisten aktualisieren

```
sudo apt update
```

Aktualisiert die Paketinformationen auf dem System.

```
pdal@apache110:~$ sudo apt update
[sudo] password for pdal:
Sorry, try again.
[sudo] password for pdal:
Hit:1 http://archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1158 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1092 kB]
Fetched 2502 kB in 3s (898 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
pdal@apache110:~$
```

Apache2 installieren

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

```
pdal@apache110:~$ sudo apt install apache2
Reading package lists... done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1t64 lib
  libldap-common libldap2 liblua5.4-0 libperl5.38t64 l
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec
  libtap-harness-archive-perl
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libap
  libgdbm-compat4t64 libldap-common libldap2 liblua5.4
0 upgraded, 19 newly installed, 0 to remove and 0 not
Need to get 11.4 MB of archives.
After this operation, 61.5 MB of additional disk space
Do you want to continue? [Y/n] y
```

Installiert den Apache2-Webserver und startet ihn automatisch. Status prüfen:

```
systemctl status apache2
```

Standard-Webverzeichnis

Nach der Installation befindet sich das **Standard-Webverzeichnis** (Document Root) hier:

/var/www/html

Dieses Verzeichnis enthält die Standard-Webseite ([index.html](#)). Aufrufbar unter <http://<server-ip>>.

Bedeutung des Webverzeichnisses

Das Webverzeichnis ist der Speicherort für Dateien, die Apache2 über HTTP ausliefert.

Beispiel:

- Datei: `/var/www/html/test.html`
 - Aufrufbar im Browser: `http://<server-ip>/test.html`

HTML-Dokumente auf dem Apache-Server verwalten

Direkt im Document Root erstellen

Kleine HTML-Dokumente oder einfache Testseiten können direkt auf dem Server erstellt werden:

- Datei in der Konsole schreiben, speichern und schließen. (z.B. Proxmox->LXC->Console oder mit SSH)
- Die Seite ist anschließend im Browser unter <http://<server-ip>/meinewebsite.html> erreichbar.

```
sudo nano /var/www/html/meinewebsite.html
```

Für größere Projekte oder mehrere Dateien

Bei umfangreicherer Webseiten oder Projekten ist es effizienter, die Dateien lokal zu entwickeln und per **FTP/SFTP** auf den Server hochzuladen:

1. FTP-Client installieren (z. B. FileZilla, WinSCP).
2. Verbindung zum Server herstellen:
 - Host: <Server-IP>
 - Benutzername und Passwort des Server-Accounts
 - Port: 22 für SFTP (sicher)
3. Lokales Projektverzeichnis auswählen.
4. Dateien in das Server-Verzeichnis </var/www/html/> hochladen.
5. Berechtigungen prüfen und ggf. anpassen:

```
sudo chown -R www-data:www-data /var/www/html/mein_projekt
sudo chmod -R 755 /var/www/html/mein_projekt
```

Die Webseite ist nach dem Hochladen und Setzen der Rechte im Browser verfügbar.

Beispielinhalt:

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1">
  <meta name="description" content="Ein W3C-konformes Beispiel-Dokument zur
Überprüfung der Apache-Konfiguration">
  <title>Apache Testseite</title>
</head>
<body>
  <header>
    <h1>Apache Testseite funktioniert!</h1>
  </header>
  <div>
    <p>Diese Seite dient als einfacher Funktionstest des Webservers.</p>
  </div>
</body>
</html>
```

```
<h2>Serverinformationen</h2>
<ul>
    <li>Document Root: <code>/var/www/html</code></li>
    <li>HTTP-Status: OK</li>
</ul>
</div>
</body>
</html>
```

Datei speichern und Editor schließen.

Testen Im Browser aufrufen:

```
http://<server-ip/meineseite.html>
```

Die Meldung "**Apache Testseite funktioniert!**" sollte erscheinen und bestätigt, dass das Webverzeichnis korrekt eingebunden ist.

2. PHP installieren und testen

Einführung PHP (Hypertext Preprocessor) ist eine serverseitige Skriptsprache, die speziell für die Webentwicklung entwickelt wurde. Sie ermöglicht es, dynamische Webseiten zu erstellen, die Inhalte abhängig von Benutzereingaben, Datenbanken oder externen Systemen generieren können.

Vorteile gegenüber statischen HTML-Seiten:

- Inhalte können automatisch aktualisiert werden (z. B. aktuelle Nachrichten, Benutzerprofile, Statistiken).
- Ermöglicht Interaktion mit Datenbanken (z. B. MySQL, PostgreSQL).
- Unterstützt Sessions und Formulare, um Benutzeranmeldungen und personalisierte Inhalte bereitzustellen.
- Reduziert den Wartungsaufwand, da Inhalte zentral verwaltet und dynamisch ausgeliefert werden können.

PHP und benötigte Module installieren

```
sudo apt update
sudo apt install php libapache2-mod-php php-cli -y
```

Installiert PHP, das Apache2-PHP-Modul und die PHP-CLI.

Apache neu laden

```
sudo systemctl reload apache2
```

PHP-Dokumente auf dem Apache-Server verwalten

PHP-Dokument direkt im Document Root erstellen

Kleine PHP-Dateien oder einfache Testskripte können direkt auf dem Server erstellt werden:

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

Beispielinhalt:

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

- Datei speichern und schließen.
- Die PHP-Seite ist anschließend im Browser unter <http://<server-ip>/phpinfo.php> erreichbar.

Testen Im Browser aufrufen:

```
http://<server-ip>/phpinfo.php
```

Die PHP-Info-Seite sollte erscheinen und Informationen zur PHP-Installation anzeigen.

Für größere PHP-Projekte oder mehrere Dateien

Bei umfangreicheren PHP-Projekten ist es effizienter, die Dateien lokal zu entwickeln und per **FTP/SFTP** auf den Server hochzuladen:

1. FTP-Client installieren (z. B. FileZilla, WinSCP).
2. Verbindung zum Server herstellen:
 - Host: <Server-IP>
 - Benutzername und Passwort des Server-Accounts
 - Port: 22 für SFTP (sicher)
3. Lokales Projektverzeichnis auswählen.
4. Dateien in das Server-Verzeichnis </var/www/html/> hochladen.
5. Berechtigungen prüfen und ggf. anpassen:

```
sudo chown -R www-data:www-data /var/www/html/mein_projekt  
sudo chmod -R 755 /var/www/html/mein_projekt
```

Die PHP-Webanwendung ist nach dem Hochladen und Setzen der Rechte im Browser verfügbar.

3. Gruppen anlegen (optional)

Auf einem Linux-Server werden Benutzergruppen verwendet, um Berechtigungen für mehrere Benutzer zentral zu verwalten. Gruppen erleichtern die Verwaltung von Datei- und Verzeichniszugriffen und bestimmen, welche Benutzer bestimmte Aktionen ausführen dürfen, ohne jedem Benutzer einzeln Rechte zu vergeben.

Warum Gruppen auf einem Apache2-Server anlegen?

1. Gruppe stud

- Diese Gruppe kann für Studenten oder Entwickler angelegt werden, die auf bestimmte Webverzeichnisse oder Dateien zugreifen sollen; insbesondere bei Zugriff von mehreren Benutzern.
- Mitglieder dieser Gruppe können dann Lese- oder Schreibrechte auf Webinhalte erhalten, ohne dass jeder Benutzer individuell angepasst werden muss.
- Beispiel: Wenn `/var/www/html/meine_webseite` der Gruppe `stud` gehört, können alle Gruppenmitglieder Änderungen an den Dateien vornehmen, während andere Benutzer nur Leserechte haben oder keinen Zugriff.

2. Gruppe apacherestart

- Diese Gruppe wird verwendet, um die Berechtigung zum Neustarten oder Neuladen von Apache zu steuern.
- Nicht jeder Benutzer sollte den Webserver neu starten oder reloaden können, da dies den laufenden Betrieb beeinflusst.
- Durch das Anlegen der Gruppe und das Hinzufügen autorisierter Benutzer zu dieser Gruppe kann die Verwaltung sicher und kontrolliert erfolgen.
- Beispiel: Mitglieder der Gruppe apacherestart können über `sudo systemctl reload apache2` oder `sudo systemctl restart apache2` den Server neu starten, ohne dass alle Benutzer Root-Rechte benötigen.

Vorteile der Gruppenverwaltung

- **Zentralisierte Rechteverwaltung:** Änderungen an Berechtigungen müssen nur einmal für die Gruppe vorgenommen werden.
- **Sicherheit:** Nur bestimmte Benutzer erhalten privilegierte Rechte (z.B. Neustart des Webservers).
- **Flexibilität:** Neue Benutzer können einfach der entsprechenden Gruppe hinzugefügt werden, ohne dass manuell Dateirechte angepasst werden müssen.

Hinweis: Die Befehle

```
sudo groupadd stud  
sudo groupadd apacherestart
```

erstellen die Gruppen. Fehlermeldungen können ignoriert werden, wenn die Gruppen bereits existieren.

```
pdal@apache110:~$ sudo groupadd stud  
[sudo] password for pdal:  
pdal@apache110:~$ sudo groupadd apacherestart
```

🔒 4. Sudo-Rechte für Apache-Dienste

Warum sudo-Rechte für Apache-Dienste einrichten?

Der Apache-Webserver wird in der Regel unter einem **nicht privilegierten Benutzer** ([www-data](#)) ausgeführt, um die Sicherheit zu erhöhen. Aktionen wie **Neustart oder Reload** des Apache-Dienstes erfordern jedoch **Root-Rechte**, da sie direkt Systemdienste beeinflussen.

Um nicht jedem Benutzer Root-Rechte zu geben, wird in diesem Fall die Gruppe [apacherestart](#) verwendet:

- Nur Mitglieder dieser Gruppe dürfen **Apache-Dienste neu starten oder reloaden**.
- Die Rechte werden über [sudo](#) gesteuert, sodass autorisierte Benutzer **temporär erhöhte Berechtigungen** für diese spezifischen Befehle erhalten.
- Dies erhöht die Sicherheit, da nur berechtigte Benutzer kritische Aktionen ausführen können, während andere Benutzer weiterhin eingeschränkten Zugriff haben.

Vorteil: Keine generellen Root-Rechte nötig, Verwaltung von Berechtigungen erfolgt zentral über die Gruppe [apacherestart](#).

Wir verwenden [visudo](#) um sudo-Rechte für bestimmte Benutzer oder Benutzergruppen ein zu richten.

Warum visudo für sudo-Rechte verwendet wird [visudo](#) ist ein **sicheres Werkzeug zum Bearbeiten der sudo-Konfigurationsdatei [/etc/sudoers](#).**

Funktionen und Vorteile von visudo:

- **Syntaxprüfung:** [visudo](#) prüft beim Speichern, ob die Datei korrekt formatiert ist. Fehler in [/etc/sudoers](#) können sonst dazu führen, dass keine sudo-Befehle mehr ausgeführt werden können.
- **Sperrmechanismus:** Nur ein Benutzer kann die Datei gleichzeitig bearbeiten, um Konflikte zu vermeiden.
- **Sichere Konfiguration:** Änderungen an Benutzer- oder Gruppenrechten werden zuverlässig übernommen, ohne das System zu gefährden.

Zusammenhang mit der Gruppe [apacherestart](#):

- Wenn sudo-Rechte für die Gruppe `apacherestart` eingerichtet werden, wird die Datei `/etc/sudoers` angepasst.
- `visudo` stellt sicher, dass die neue Regel korrekt implementiert wird, sodass Gruppenmitglieder Apache-Dienste mit `sudo systemctl restart apache2` oder `sudo systemctl reload apache2` ausführen können, ohne Root-Rechte für das gesamte System zu erhalten.

```
sudo visudo
```

```
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/s

# This fixes CVE-2005-4890 and possibly breaks some versions of kdesu
# (#1011624, https://bugs.kde.org/show_bug.cgi?id=452532)
Defaults      use_pty

# This preserves proxy settings from user environments of root
# equivalent users (group sudo)
#Defaults:@sudo env_keep += "http_proxy https_proxy ftp_proxy all_proxy no_proxy"

# This allows running arbitrary commands, but so does ALL, and it means
# different sudoers have their choice of editor respected.
#Defaults:@sudo env_keep += "EDITOR"

# Completely harmless preservation of a user preference.
#Defaults:@sudo env_keep += "GREP_COLOR"

# While you shouldn't normally run git as root, you need to with etckeeper
#Defaults:@sudo env_keep += "GIT_AUTHOR_* GIT_COMMITTER_"

# Per-user preferences; root won't have sensible values for them.
```

am Ende der Datei folgende Zeilen hinzufügen

```
%apacherestart ALL=(ALL) NOPASSWD: /bin/systemctl reload apache2
%apacherestart ALL=(ALL) NOPASSWD: /bin/systemctl restart apache2
```

```
# Allow members of group sudo to execute any command
@sudo  ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d
%apacherestart ALL=(ALL) NOPASSWD: /bin/systemctl reload apache2
%apacherestart ALL=(ALL) NOPASSWD: /bin/systemctl restart apache2
```

[Wrote 59 lines]

abspeichern mit `Strg + S` gefolgt von `Enter`. Jetzt den Editor mit `Strg + X` schließen.

👤 5. Benutzer user1 anlegen

In diesem Schritt wird ein neuer Benutzer **user1** erstellt, ihm ein Home-Verzeichnis und die Bash-Shell zugewiesen, das Passwort gesetzt und er wird sowohl der Gruppe **stud** für Zugriff auf Webinhalte als auch der Gruppe **apacherestart** für das Neustarten oder Reloaden von Apache-Diensten hinzugefügt.

```
sudo useradd -m -d /home/user1 -s /bin/bash -g stud user1
echo "user1:meinpasswort" | sudo chpasswd
sudo usermod -aG apacherestart user1
```

```
pdal@apache110:~$ sudo useradd -m -d /home/user1 -s /bin/bash -g stud user1
[sudo] password for pdal:
pdal@apache110:~$ echo "user1:JadeHS20" | sudo chpasswd
pdal@apache110:~$ sudo usermod -aG apacherestart user1
pdal@apache110:~$ █
```

📁 6. Web-Verzeichnis für user1 einrichten

In diesem Schritt wird ein persönliches Web-Verzeichnis für den Benutzer **user1** erstellt, die Besitzrechte auf **user1** und die Gruppe **stud** gesetzt und die Zugriffsrechte des Home-Verzeichnisses so angepasst, dass der Benutzer auf seine Webinhalte zugreifen kann. Der Standard-User hat keine Schreibrechte auf das Standard-Apache2-Webverzeichnis **/var/www/html/**. Aus diesem Grund erstellt man im **/home**-Verzeichnis des Users, ein Webverzeichnis, das in die Apache-Konfiguration eingebunden wird.

```
sudo mkdir -p /home/user1/www
sudo chown user1:stud /home/user1/www
sudo chmod 755 /home/user1
```

```
pdal@apache110:~$ sudo mkdir -p /home/user1/www
pdal@apache110:~$ sudo chown user1:stud /home/user1/www
pdal@apache110:~$ sudo chmod 755 /home/user1
pdal@apache110:~$ █
```

🔗 7. Apache Alias-Konfiguration für user1

Ziel

Einrichtung eines Alias-Verzeichnisses für einen bestimmten Benutzer (**user1**), damit der Inhalt des Verzeichnisses **/home/user1/www** über den Webbrowser erreichbar ist.

Neue Apache-Konfigurationsdatei erstellen

Man öffnet einen Editor seiner Wahl (z. B. **nano**) und erstellt die Datei:

```
sudo nano /etc/apache2/sites-available/stud-aliases.conf
```

Jetzt fügt man den nachfolgenden Inhalt in die Datei ein.

```
<IfModule alias_module>
    Alias /user1 /home/user1/www
    <Directory /home/user1/www>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</IfModule>
```

Speichern mit **STRG+O**, bestätigen mit Enter, und Editor schließen mit **STRG+X**.

```
webmin ALL=(ALL) NOPASSWD: /usr/bin/apt, /usr/bin/apt-get, /usr/bin/apt-cache, /bin/systemctl
pdal@apache110:~$ sudo tee /etc/apache2/sites-available/stud-aliases.conf > /dev/null <<EOF
<IfModule alias_module>
Alias /user1 /home/user1/www
<Directory /home/user1/www>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
</IfModule>
EOF
[sudo] password for pdal:
pdal@apache110:~$
```

⑧ Apache-Site aktivieren & reload

```
sudo a2ensite stud-aliases.conf
sudo systemctl reload apache2
```

```
pdal@apache110:~$ sudo a2ensite stud-aliases.conf
Enabling site stud aliases.
To activate the new configuration, you need to run:
    systemctl reload apache2
pdal@apache110:~$ sudo systemctl re
reboot          reenable           reload
pdal@apache110:~$ sudo systemctl reload apache2.service
pdal@apache110:~$
```

Ergebnis

- Apache2 ist installiert
- Benutzer user1 hat ein Webverzeichnis unter /home/user1/www erreichbar über:

```
http://<IP-des-Containers>/user1
```

user1 kann Apache reloaden, z. B. mit:

```
sudo systemctl reload apache2
```

☞ Tipps zur Prüfung Webzugriff testen:

```
curl http://localhost/user1
```

Apache-Status:

```
systemctl status apache2
```

Quellen

- „Dokumentation zum Apache HTTP Server Version 2.4 - Apache HTTP Server Version 2.4“. Zugegriffen: 22. September 2025. [Online]. Verfügbar unter: [apache doc](#)
- „Access Control - Apache HTTP Server Version 2.4“. Zugegriffen: 24. September 2025. [Online]. Verfügbar unter: [Apache2 Access Control](#)
- „Apache HTTP Server Tutorial: .htaccess files - Apache HTTP Server Version 2.4“. Zugegriffen: 24. September 2025. [Online]. Verfügbar unter: [Apache2 htaccess](#)
- „Apache httpd Tutorial: Introduction to Server Side Includes - Apache HTTP Server Version 2.4“. Zugegriffen: 24. September 2025. [Online]. Verfügbar unter: [Apache2 SSI](#)
- „Apache SSL/TLS Encryption - Apache HTTP Server Version 2.4“. Zugegriffen: 24. September 2025. [Online]. Verfügbar unter: [Apache2 SSL/TLS](#)
- „Apache starten - Apache HTTP Server Version 2.4“. Zugegriffen: 24. September 2025. [Online]. Verfügbar unter: [Apache2 starten](#)
- „Apache Tutorial: Dynamic Content with CGI - Apache HTTP Server Version 2.4“. Zugegriffen: 24. September 2025. [Online]. Verfügbar unter: [Apache2 CGI](#)
- „Authentication and Authorization - Apache HTTP Server Version 2.4“. Zugegriffen: 24. September 2025. [Online]. Verfügbar unter: [Apache2 Authentication](#)
- „Server-Wide Configuration - Apache HTTP Server Version 2.4“. Zugegriffen: 24. September 2025. [Online]. Verfügbar unter: [Apache2 Server Konfiguration](#)
- „Konfigurationsdateien - Apache HTTP Server Version 2.4“. Zugegriffen: 24. September 2025. [Online]. Verfügbar unter: [Apache2 Konfigurationsdateien](#)
- „Getting Started - Apache HTTP Server Version 2.4“. Zugegriffen: 24. September 2025. [Online]. Verfügbar unter: [Apache2 getting started](#)
- „Befehlsübersicht > Shell > Wiki > ubuntuusers.de“. Zugegriffen: 20. August 2025. [Online]. Verfügbar unter: [Shell Befehle](#)

Lizenz

Dieses Werk ist lizenziert unter der **Creative Commons - Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz**.

[Zum Lizenztext auf der Creative Commons Webseite](#)