Piwigo auf Unraid

Klicke hier, um den großen Block zu öffnen

Piwigo ist eine hervorragende Open-Source-Bilddatenbank, mit der Handwerksbetriebe ihre Baustellendokumentation, Projektbilder und Kunden-Referenzen effizient und datenschutzkonform organisieren können

Da Handwerker täglich unzählige Fotos von Baufortschritten, Mängeln und fertigen Gewerken machen, dient Piwigo als zentrales, digitales Archiv.

Vorteile für Handwerker

- **Strukturierte Baustellendokumentation:** Alben lassen sich flexibel nach Kundenname, Projektnummer oder Jahr sortieren. Mithilfe von Schlagworten (Tags) filterst du blitzschnell nach „Rohbau“, „Sanitär“, „Mangel“ oder „Vorher-Nachher“.
- **Rechtssicherheit & Nachweis:** Fotos sicherst du gegen unberechtigte Mängelansprüche ab. Über den Datumsfilter findest du exakt das Foto vom Zustand an Tag X.
- **Referenzmappen für Kunden:** Du kannst ausgewählte Alben für potenzielle Kunden freigeben, um eure Arbeit digital zu präsentieren (z. B. auf einem Tablet direkt beim Beratungsgespräch).
- **Volle Datenkontrolle & DSGVO:** Im Gegensatz zu US-Cloud-Diensten kannst du Piwigo auf deinem eigenen Webserver oder Firmen-NAS installieren (Self-Hosting). Deine Kundendaten und Baustellenfotos bleiben komplett in deiner Hand.
- **Mobiles Hochladen per App:** Mitarbeiter können Fotos direkt von der Baustelle per Smartphone-App in die Datenbank hochladen.
- **Rechteverwaltung:** Du entscheidest, welche Mitarbeiter Zugriff auf welche Projektordner haben.

Schritt für Schritt zu einer eigenen Foto- und Video-Galerie: MariaDB und Piwigo als Container, Video-Wiedergabe aktiviert, Upload-Limits hochgesetzt und Alben, die sich zum Präsentieren eignen.

`Docker · LinuxServer.io Image · MariaDB · Unraid 7.3.2 · Fotos + Videos`

00 · Überblick

Piwigo ist eine quelloffene Software für eine eigene, selbst gehostete Foto- und Video-Galerie. Damit organisierst du Bilder und Videos in Alben, versiehst sie mit Tags und Beschreibungen und gibst sie gezielt frei: öffentlich, nur für eingeloggte Nutzer oder passwortgeschützt für einzelne Alben. Typische Einsatzzwecke sind private Familien- und Urlaubsalben, Vereins- oder Veranstaltungsfotos sowie Portfolios für Kunden, ohne dabei auf Google Fotos, Dropbox & Co. angewiesen zu sein.

Piwigo braucht zwei Bausteine: einen **Piwigo-Container** (Webserver + PHP, Bild von LinuxServer.io) und eine **MariaDB-Datenbank**, in der Alben, Nutzer und Metadaten liegen. Die eigentlichen Fotos und Videos landen auf einem eigenen Unraid-Share, nicht in der Datenbank.

```
[ Browser / Mobile-App ]
          |
          |  Upload & Betrachten
          v
[ Piwigo-Container ]  :80 · PHP + Nginx
          |
          +---> [ MariaDB-Container ]  :3306 · Alben, Nutzer, Metadaten
          |
          +---> [ Share /gallery ]     Original-Fotos & -Videos
```

Das Ergebnis: eine Galerie mit Alben, feiner Rechteverwaltung (z. B. passwortgeschützte Familienalben) und, nach ein paar Zusatzschritten, sauber abspielbaren Videos.

01 · Voraussetzungen

- Unraid mit installiertem **Community Applications**-Plugin.
- Zwei Shares bzw. Ordner planen: einen für die App-Konfiguration (z. B. `appdata/piwigo`) und einen eigenen für die Mediendateien (z. B. `gallery`), getrennt von `appdata`, damit große Videos nicht im appdata-Backup landen.
- PUID/PGID: Unraid nutzt standardmäßig den Benutzer `nobody` (UID 99) und die Gruppe `users` (GID 100), nicht die `1000/1000`, die in der allgemeinen LinuxServer-Doku auftauchen. Für Unraid `PUID=99` und `PGID=100` verwenden.

> **Tipp:** Vergib der Unraid-Weboberfläche und den Containern eine feste IP oder nutze durchgehend Container-*Namen* statt IP-Adressen. Das erspart dir Nacharbeit, falls sich die DHCP-Adresse später ändert (siehe Abschnitt 09 · Fehlerbehebung).

02 · Datenbank (MariaDB)

Läuft auf deinem Unraid bereits ein Datenbank-Container und Port `3306` ist schon belegt, gibt es zwei sinnvolle Wege. Beide sind hier beschrieben.

Option A · vorhandenen Container mitbenutzen (empfohlen)

Piwigo braucht keinen eigenen Datenbankserver, nur eine eigene Datenbank und einen eigenen Benutzer darin. Das spart RAM und einen weiteren Container. Direkt in der laufenden Datenbank anlegen:

```
docker exec -it <name-deines-db-containers> mysql -u root -p
```

> **Alternative:** Genauso gut geht das über das **Konsolen-Icon** („>_„) des DB-Containers im Unraid-WebUI (Docker-Tab). Das öffnet direkt eine Shell *innerhalb* des Containers. Dort reicht dann einfach `mysql -u root -p`, ohne das vorangestellte `docker exec -it …`. Kein SSH auf den Unraid-Host nötig.

Danach in der MySQL-Shell (Container- und Nutzernamen anpassen):

mysql-shell

```
CREATE DATABASE piwigo CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
CREATE USER 'piwigo'@'%' IDENTIFIED BY 'DEIN_PASSWORT';
GRANT ALL PRIVILEGES ON piwigo.* TO 'piwigo'@'%';
FLUSH PRIVILEGES;
EXIT;
```

> **Per Adminer statt Kommandozeile?** Wirfst du diesen Block in Adminers SQL-Befehl-Funktion statt in die `mysql`-Shell: Bei `CREATE USER`/`GRANT`/`FLUSH` ist „0 Datensätze betroffen„ normal, kein Fehler. Die letzte Zeile `EXIT;` dagegen lass weg: Das ist ein Kommando nur für die `mysql`-Kommandozeile, kein echtes SQL, und erzeugt in Adminer einen harmlosen Syntaxfehler.

Alternative per Weboberfläche: Adminer

Wer die drei SQL-Zeilen lieber klickt statt tippt: Läuft noch kein Verwaltungstool für die Datenbank, lohnt sich die Installation von **Adminer**, einem einzigen PHP-Skript ohne eigene Konfiguration, das in Sekunden startklar ist.

docker-compose.yml

```
services:
  adminer:
    image: adminer:latest
    container_name: adminer
    restart: unless-stopped
    environment:
      - ADMINER_DEFAULT_SERVER=<name-oder-ip-deines-db-containers>
    ports:
      - 8087:8080
```

Alternativ genauso einfach über Community Applications: nach `Adminer` suchen und installieren. Danach im Browser öffnen, als System `MySQL/MariaDB` wählen, Server/Host sowie Benutzername `root` und Root-Passwort eingeben.

> **Wichtig beim Login:** Das Feld „Datenbank„ im Login-Formular **leer lassen**: `piwigo` existiert ja noch nicht, ein Login direkt damit schlägt fehl. Ohne dieses Feld landest du auf Server-Ebene mit der Liste aller Datenbanken.

- **Datenbank anlegen:** Auf Server-Ebene unten auf *„Create database„* klicken, Namen `piwigo` und Kollation `utf8mb4_unicode_ci` eintragen.
- **Benutzer anlegen:** Auf Server-Ebene (oder in der neuen Datenbank) den Menüpunkt *„Privileges“* öffnen, dort *„Create user„* wählen und Benutzername, Passwort sowie die Rechte auf `piwigo` vergeben.

Das entspricht genau den drei SQL-Zeilen von oben, nur per Klick statt getippt.

> **Adminer oder phpMyAdmin?** Für diese eine Aufgabe, eine Datenbank plus einen Benutzer anlegen, ist Adminer klar die bessere Wahl: eine einzelne, kleine PHP-Datei statt einer vollständigen Anwendung mit vielen Dateien, dadurch schneller geladen und mit kleinerer Angriffsfläche. phpMyAdmin hat mehr Funktionstiefe für komplexere Datenbankarbeit (feingranulare Rechteverwaltung, mehr Import/Export-Optionen), für einen schnellen CREATE-DATABASE-Vorgang im Homelab ist das aber nicht nötig.

Als Datenbank-Host trägst du im Piwigo-Assistenten die IP des Unraid-Servers (oder den Container-Namen, in einem gemeinsamen Docker-Netzwerk) mit dem **gewohnten Port** ein, z. B. `3306`. Der ändert sich nicht, du legst nur eine zusätzliche Datenbank in der bestehenden Instanz an.

> **Geteilter Container:** Ein Image-Update des DB-Containers betrifft die Daten nicht: Datenbank, Benutzer und Rechte liegen im gemappten Volume, nicht im Image. Was sich ändert: Piwigo teilt sich jetzt Wartungsfenster und kurze Neustart-Ausfälle mit der anderen App. Vor größeren Versionssprüngen lohnt sich ein Dump **aller** Datenbanken (`mysqldump –all-databases`), nicht nur von `piwigo`.

Option B · eigener, zweiter Datenbank-Container

Spricht etwas dagegen, den vorhandenen Server mitzubenutzen (andere App, andere Backup-Routine, härtere Trennung gewünscht), bekommt Piwigo seine eigene MariaDB, nur eben nicht auf Host-Port 3306, der ist ja schon vergeben. Der Container selbst lauscht *innerhalb* weiterhin ganz normal auf 3306, nur die Weiterleitung nach außen bekommt eine andere Nummer, z. B. `3307`:

docker-compose.yml

```yaml
services:
  piwigo-db:
    image: lscr.io/linuxserver/mariadb:latest
    container_name: piwigo-db
    environment:
      - PUID=99
      - PGID=100
      - TZ=Europe/Berlin
      - MYSQL_ROOT_PASSWORD=ein-sicheres-passwort
      - MYSQL_DATABASE=piwigo
      - MYSQL_USER=piwigo
      - MYSQL_PASSWORD=eigenes-passwort
    volumes:
      - /mnt/user/appdata/piwigo-db:/config
    ports:
      - 3307:3306
    restart: unless-stopped
```

Im Piwigo-Setup-Assistenten dann als Datenbank-Host `<Unraid-IP>:3307` eintragen: Host und Port zusammen, getrennt durch Doppelpunkt. Hängst du `piwigo` und `piwigo-db` stattdessen in ein gemeinsames, benutzerdefiniertes Docker-Netzwerk, reicht als Host der Container-Name `piwigo-db` mit dem **internen** Port `3306`. Die externe Portverschiebung betrifft dann nur den Zugriff von

außerhalb des Netzwerks.

> **Hinweis zur Adresse:** Egal für welche Option: Wer beide Container in ein gemeinsames Docker-Netzwerk hängt, kann durchgehend mit **Container-Namen** statt IP-Adressen arbeiten. Das übersteht spätere IP-Änderungen des Unraid-Servers.

03 · Piwigo-Container

In Community Applications nach `Piwigo` suchen. Ist kein Template gelistet, lässt sich der Container auch manuell über „Docker → Add Container“ mit dem Repository `lscr.io/linuxserver/piwigo` anlegen.

docker-compose.yml

```yaml
services:
  piwigo:
    image: lscr.io/linuxserver/piwigo:latest
    container_name: piwigo
    environment:
      - PUID=99
      - PGID=100
      - TZ=Europe/Berlin
    volumes:
      - /mnt/user/appdata/piwigo:/config
      - /mnt/user/gallery:/gallery
    ports:
      - 8181:80
    restart: unless-stopped
```

Wichtig sind die beiden Volumes: `/config` für die Piwigo-Installation selbst, `/gallery` für die eigentlichen Fotos und Videos. Den Host-Port `8181` kannst du frei wählen, falls `80` auf deinem Unraid schon belegt ist.

04 · Ersteinrichtung

Container starten und im Browser http://<Unraid-IP>:8181 öffnen. Der Setup-Assistent fragt nach:

- **Datenbank-Host:** IP des Unraid-Servers (oder Container-Name, siehe oben)
- **Datenbank-Name / Benutzer / Passwort:** die Werte aus Schritt 02
- **Admin-Konto:** Benutzername, Passwort, E-Mail-Adresse für dich selbst
- **Galerie-Name:** Titel, der oben in der Galerie erscheint

Nach dem Assistenten landest du direkt im Admin-Bereich. Von hier aus geht es an Videos und Uploads.

05 · Video-Wiedergabe aktivieren

Piwigo nimmt Videos zwar entgegen, spielt sie aber ohne Zusatz-Plugin nicht im Browser ab.

Video-Plugin installieren: Administration → Module → Erweiterungen: nach `VideoJS` suchen und installieren. Taucht das Plugin unter diesem Namen nicht mehr auf, nach `piwigo-videojs` suchen, so heißt die Erweiterung inzwischen. Erst nach der Aktivierung werden Videos inline abgespielt statt nur als Datei-Download angeboten.

Erlaubte Dateiformate: Unter Administration → Konfiguration → Optionen die erlaubten Datei-Endungen um die gängigen Web-Formate erweitern:

Empfohlen	`mp4`, `m4v`, `webm`, `ogv`
Vermeiden	`mov`: iPhone-Videos vorher nach MP4 (H.264 + AAC) konvertieren, sonst spielen viele Browser sie nicht ab

> **Achtung:** Für automatisch erzeugte Video-Vorschaubilder braucht Piwigo zusätzlich `ffmpeg` im Container (im LinuxServer-Image i. d. R. enthalten) sowie eines von `ffprobe`, `exiftool` oder `MediaInfo` für Metadaten. Fehlen Vorschaubilder, zuerst hier ansetzen.

06 · Upload-Limits erhöhen

PHP begrenzt Uploads standardmäßig auf wenige MB: für Fotos meist ausreichend, für Videos schnell zu wenig.

Zuerst prüfen, welche Konfigurationsdatei im Container existiert:

```
docker exec -it piwigo find / -maxdepth 4 -iname "php-local.ini" -o -iname "php.ini" 2>/dev/null
```

Bei LinuxServer-PHP-Images liegt eine überschreibbare `php-local.ini` meist unterhalb von `/config`. Dort (oder in der gefundenen Datei) ergänzen:

php-local.ini

```
upload_max_filesize = 2048M
post_max_size = 2048M
memory_limit = 512M
max_execution_time = 600
```

Anschließend den Container neu starten. Läuft davor ein **Reverse Proxy** (SWAG, Nginx Proxy Manager), muss dessen Body-Size-Limit (z. B. `client_max_body_size`) ebenfalls angehoben werden, sonst bricht der Upload schon dort ab, bevor er Piwigo überhaupt erreicht.

> **Praxiswert:** 2 GB pro Datei ist für die meisten Handy-Videos großzügig bemessen. Wer regelmäßig größere Rohdateien hochlädt, setzt die Werte entsprechend höher.

07 · Alben & Präsentation

- **Struktur zuerst:** Alben und Unteralben anlegen, bevor der große Upload beginnt, denn Piwigo sortiert nachträglich verschobene Fotos nicht automatisch neu ein.
- **Upload:** per Weboberfläche (Mehrfachauswahl) oder über die offizielle Mobile-App direkt vom Smartphone in ein bestehendes Album.
- **Diashow:** Piwigo bringt eine eingebaute Diashow-Ansicht pro Album mit; für größere Präsentationen lohnt ein Blick in die Theme-Galerie (Administration → Module → Designs) für eine ruhigere, bildlastigere Darstellung.
- **Freigabe ohne Benutzerkonto:** Alben lassen sich mit einem Album-Passwort versehen und der Link direkt teilen, praktisch für Familie oder Kunden, ohne dass jemand ein eigenes Konto braucht.

08 · Von unterwegs erreichen (optional)

Für Zugriff außerhalb des Heimnetzes nicht den Port direkt am Router freigeben, sondern einen **Reverse Proxy mit HTTPS** davorschalten (z. B. SWAG oder Nginx Proxy Manager, jeweils ebenfalls als Unraid-Container). Piwigo bekommt dann eine eigene Subdomain, der Proxy übernimmt Zertifikat und Verschlüsselung.

> **Sicherheit:** Starkes Admin-Passwort, regelmäßige Updates des Piwigo-Containers und, falls von außen erreichbar, ein Zwei-Faktor-Login-Plugin in Betracht ziehen.

09 · Fehlerbehebung

Symptom	Ursache & Lösung
Access denied for user „piwigo“@„…“	Benutzerrechte in MariaDB prüfen, oder Datenbank-Host im Piwigo-Setup korrigieren (`config/database.inc.php`). Container-IP statt Name kann sich nach Neustart ändern.
Piwigo erreicht die Datenbank nach Unraid-Neustart nicht mehr	Meist eine geänderte Container-IP. Feste IP vergeben oder auf ein benutzerdefiniertes Docker-Netzwerk mit Container-Namen umstellen.
Video wird als Download angeboten statt abgespielt	Video-Plugin (VideoJS bzw. piwigo-videojs) fehlt oder ist deaktiviert (Schritt 05).
Video wird gar nicht erst akzeptiert	Dateiendung nicht freigegeben, oder es handelt sich um eine `.mov`-Datei. Vorher nach MP4 konvertieren.
Upload großer Dateien bricht ab	`upload_max_filesize`/`post_max_size` in der php-local.ini zu niedrig (Schritt 06), oder Limit im vorgeschalteten Reverse Proxy.
Rechte-/Schreibfehler beim Hochladen	PUID/PGID des Containers passen nicht zu den Berechtigungen des `gallery`-Shares. Auf Unraid `99/100` verwenden.
Host-Port `3306` bereits belegt	Vorhandenen DB-Container mitbenutzen (Option A) oder neuen Container auf einen freien Host-Port wie `3307` legen (Option B). Siehe Abschnitt 02.

10 · Backup

Drei Dinge gehören gesichert, damit im Notfall alles wiederherstellbar ist:

- `/mnt/user/appdata/piwigo`: Konfiguration und Zugangsdaten
- `/mnt/user/gallery`: die Original-Fotos und -Videos selbst
- ein regelmäßiger **MariaDB-Dump** der `piwigo`-Datenbank (Alben, Nutzer, Metadaten stecken hier, nicht in den Dateien)

```
docker exec mariadb mysqldump -u piwigo -p piwigo > piwigo-backup-$(date +%F).sql
```

Das CA-Plugin „Appdata Backup / Restore„ deckt den ersten Punkt automatisiert ab; Datenbank-Dump und Gallery-Share sollten zusätzlich in die eigene Backup-Routine (z. B. auf ein zweites Ziel) aufgenommen werden.

Produktiv eingerichtet und getestet mit dem LinuxServer.io-Image `lscr.io/linuxserver/piwigo` und MariaDB auf Unraid 7.3.2. Piwigo- und Docker-Versionsstände ändern sich, Menüpfade können in neueren Versionen leicht abweichen.