

Psono Passwortmanager

Neuinstallation — Schritt-für-Schritt-Anleitung

Stand: Juni 2026 · Psono Community Edition · Debian 12/13 · PostgreSQL 17 · Nginx · Dehydrated

Platzhalter in dieser Anleitung

Ersetze alle Platzhalter vor dem ersten Schritt durch deine eigenen Werte:

Platzhalter	Bedeutung	Beispiel
DEINE_DOMAIN	Öffentliche Domain von Psono	psono.example.com
DEIN_SERVER_HOSTNAME	Hostname des Servers	server.example.com
DEINE_SERVER_IP	Öffentliche IPv4 des Servers	203.0.113.42
DEIN_DB_PASSWORT	Passwort für PostgreSQL-Benutzer psono	sicheres-passwort-hier
DEINE_EMAIL	E-Mail für Psono-Benachrichtigungen	admin@example.com
DEIN_SMTP_PASSWORT	Passwort für den SMTP-Server	smtp-passwort-hier

Überblick & Voraussetzungen

Diese Anleitung führt durch eine komplette Erstinstallation von Psono Community Edition auf einem frischen Debian-Server.

Was bereits vorhanden sein muss

- ✓ Debian 12 oder 13 mit root-Zugriff
- ✓ Docker installiert (empfohlen: download.docker.com, nicht das Debian-Paket)
- ✓ PostgreSQL 17 installiert
- ✓ Nginx installiert
- ✓ Dehydrated installiert mit gültigem Zertifikat für DEINE_DOMAIN
- ✓ DNS-Eintrag für DEINE_DOMAIN zeigt auf DEINE_SERVER_IP
- ✓ SMTP-Zugangsdaten für den E-Mail-Versand (Aktivierungs-Mails)

Übersicht der Phasen

1

PostgreSQL vorbereiten

Datenbank, Benutzer, Erweiterungen und Netzwerkkonfiguration

2

Psono-Konfiguration erstellen

Kryptographische Schlüssel generieren und settings.yaml anlegen

3

Psono-Container starten

Docker Compose einrichten und Container starten

4

Nginx Reverse Proxy einrichten

HTTPS-Konfiguration mit Dehydrated-Zertifikat

5

Ersteinrichtung abschliessen

Admin-Benutzer anlegen und Backups einrichten

Phase 1 — PostgreSQL vorbereiten

PostgreSQL 17 ist bereits installiert. Wir legen die Datenbank und den Psono-Benutzer an, aktivieren die benötigten Erweiterungen und konfigurieren die Netzwerkzugänge für Docker.

Schritt 1.1 — Als postgres-Benutzer anmelden

```
sudo -i -u postgres
```

Erwarteter Prompt: postgres@DEIN_SERVER_HOSTNAME:~\$

Schritt 1.2 — PostgreSQL-Konsole öffnen

```
psql
```

Erwarteter Prompt: postgres=#

Schritt 1.3 — Datenbank, Benutzer und Rechte anlegen

Ersetze DEIN_DB_PASSWORT durch ein starkes Passwort und notiere es.

```
CREATE DATABASE psono;  
CREATE USER psono WITH PASSWORD 'DEIN_DB_PASSWORT';  
GRANT ALL PRIVILEGES ON DATABASE psono TO psono;  
\c psono  
GRANT ALL ON SCHEMA public TO psono;
```



Hinweis zu \c psono

Nach \c psono wechselt der Prompt auf psono=# — das ist korrekt.

Schritt 1.4 — Erweiterungen installieren

Psono benötigt zwei PostgreSQL-Erweiterungen (Prompt ist psono=#):

```
CREATE EXTENSION IF NOT EXISTS ltree;  
CREATE EXTENSION IF NOT EXISTS pgcrypto;
```

Schritt 1.5 — psql beenden

```
\q  
exit
```

Schritt 1.6 — pg_hba.conf für lokalen psono-Benutzer anpassen

Zuerst den Pfad ermitteln:

```
sudo -u postgres psql -c "SHOW hba_file;"
```

Typischer Pfad: /etc/postgresql/17/main/pg_hba.conf. Folgende Zeile einfügen:

```
sudo sed -i '/^local    all                                all/i local    all                                psono  
md5' /etc/postgresql/17/main/pg_hba.conf
```

Schritt 1.7 — pg_hba.conf für Docker-Netzwerke anpassen

```
echo "host    psono    psono    172.17.0.0/16    md5" >>  
/etc/postgresql/17/main/pg_hba.conf  
echo "host    psono    psono    172.18.0.0/16    md5" >>  
/etc/postgresql/17/main/pg_hba.conf
```



Warum zwei Netzwerke?

172.17.0.0/16 ist das Docker-Standard-Bridge-Netzwerk. 172.18.0.0/16 legt Docker Compose automatisch für den Stack an. Der Container kommt typischerweise aus 172.18.x.x.

Schritt 1.8 — PostgreSQL auf Docker-Interfaces lauschen lassen

Standardmäßig lauscht PostgreSQL nur auf localhost. Die Docker-Bridge-IPs explizit hinzufügen:

```
sudo -u postgres psql -c "ALTER SYSTEM SET listen_addresses =  
'localhost,172.17.0.1,172.18.0.1';"
```



Bewusst eingeschränkt

Wir setzen nur die konkreten Docker-Bridge-IPs, nicht "*". So ist PostgreSQL nicht auf allen Interfaces erreichbar — nur lokal und für Docker.

Schritt 1.9 — PostgreSQL neu starten

Ein Neustart (nicht nur reload) ist nötig damit listen_addresses wirksam wird:

```
systemctl restart postgresql
```

✓ Prüfung Phase 1

Erweiterungen prüfen:

```
sudo -u postgres psql -U psono -d psono -W -c '\dx'
```

Erwartung: Tabelle mit ltree und pgcrypto. listen_addresses prüfen:

```
sudo -u postgres psql -c "SHOW listen_addresses;"
```

Erwartung: localhost,172.17.0.1,172.18.0.1

Phase 2 — Psono-Konfiguration erstellen

Psono benötigt kryptographische Schlüssel die einmalig generiert werden. Diese Schlüssel sind an die Datenbank gebunden und dürfen danach nie mehr geändert werden.

Schritt 2.1 — Verzeichnisse anlegen

```
mkdir -p /opt/docker/psono  
mkdir -p /opt/docker/psono-client
```

Schritt 2.2 — Kryptographische Schlüssel generieren

```
docker run --rm -ti psono/psono-combo:latest python3 ./psono/manage.py  
generateserverkeys
```

Die Ausgabe enthält SECRET_KEY, ACTIVATION_LINK_SECRET, DB_SECRET, EMAIL_SECRET_SALT, PRIVATE_KEY und PUBLIC_KEY. Diese Werte sofort notieren und sicher aufbewahren.

🔑 Schlüssel sicher aufbewahren

Diese Schlüssel verschlüsseln alle gespeicherten Passwörter. Gehen sie verloren, sind alle Tresore unwiederbringlich verloren. Schlüssel an einem sicheren Ort speichern.

Schritt 2.3 — settings.yaml erstellen

Erstelle /opt/docker/psono/settings.yaml und ersetze alle Platzhalter mit den generierten Schlüsseln:

```

cat > /opt/docker/psono/settings.yaml << 'EOF'
# Schlüssel aus generateserverkeys eintragen:
SECRET_KEY: 'GENERIERTER_SECRET_KEY'
ACTIVATION_LINK_SECRET: 'GENERIERTER_ACTIVATION_LINK_SECRET'
DB_SECRET: 'GENERIERTER_DB_SECRET'
EMAIL_SECRET_SALT: 'GENERIERTER_EMAIL_SECRET_SALT'
PRIVATE_KEY: 'GENERIERTER_PRIVATE_KEY'
PUBLIC_KEY: 'GENERIERTER_PUBLIC_KEY'

DEBUG: False

ALLOWED_HOSTS: ['DEINE_DOMAIN']
HOST_URL: 'https://DEINE_DOMAIN/server'
WEB_CLIENT_URL: 'https://DEINE_DOMAIN'

DATABASES:
  default:
    ENGINE: 'django.db.backends.postgresql_psycopg2'
    NAME: 'psono'
    USER: 'psono'
    PASSWORD: 'DEIN_DB_PASSWORT'
    HOST: 'host.docker.internal'
    PORT: '5432'

CACHE_ENABLE: FALSE

EMAIL_FROM: 'Psono <DEINE_EMAIL>'
EMAIL_HOST: 'smtp.example.com'
EMAIL_HOST_USER: 'DEINE_EMAIL'
EMAIL_HOST_PASSWORD: 'DEIN_SMTP_PASSWORT'
EMAIL_PORT: 587
EMAIL_USE_TLS: True

ALLOW_REGISTRATION: True
ALLOW_LOST_PASSWORD: True
EOF

```

Schritt 2.4 — config.json für den Web-Client anlegen

```

cat > /opt/docker/psono-client/config.json << 'EOF'
{
  "backend_servers": [{
    "title": "Psono Server",
    "url": "https://DEINE_DOMAIN/server"
  }],
  "base_url": "https://DEINE_DOMAIN/",
  "allow_custom_server": true,
  "allow_registration": true,
  "allow_lost_password": true,
  "disable_download_bar": false,
  "remember_me_default": false,
  "trust_device_default": false,
  "authentication_methods": ["AUTHKEY"]
}
EOF

```

Phase 3 — Psono-Container starten

Schritt 3.1 — Docker Compose Datei erstellen

```
cat > /opt/docker/psono/docker-compose.yml << 'EOF'
services:
  psono-combo:
    image: psono/psono-combo:latest
    restart: unless-stopped
    ports:
      - "127.0.0.1:10200:80"
    volumes:
      - /opt/docker/psono/settings.yaml:/root/.psono_server/settings.yaml:ro
      - /opt/docker/psono-client/config.json:/usr/share/nginx/html/config.json:ro
    extra_hosts:
      - "host.docker.internal:host-gateway"
EOF
```

Schritt 3.2 — Container starten

```
cd /opt/docker/psono
docker compose up -d
```

Schritt 3.3 — Logs prüfen

```
docker compose logs psono-combo 2>&1 | tail -20
```

Erwartung: "Listening on TCP address 0.0.0.0:8000" ohne ERROR-Meldungen.

✓ Prüfung Phase 3

```
curl -s http://localhost:10200/server/info/ | head -c 200
```

Erwartung: JSON-Antwort mit {"info": ...} und der aktuellen Psono-Version.

Phase 4 — Nginx Reverse Proxy einrichten

Schritt — Voraussetzungen prüfen

Folgendes wird vorausgesetzt:

- Dehydrated ist installiert und konfiguriert
- DEINE_DOMAIN ist in /etc/dehydrated/domains.txt eingetragen

- Die Zertifikate wurden bereits generiert:

```
ls /var/lib/dehydrated/certs/DEINE_DOMAIN/  
# Erwartung: fullchain.pem und privkey.pem
```

Schritt — Nginx-Konfiguration erstellen

Erstelle /etc/nginx/sites-available/DEINE_DOMAIN.conf (DEINE_DOMAIN ersetzen):

```
server {  
    listen 80;  
    listen [::]:80;  
    server_name DEINE_DOMAIN;  
    server_tokens off;  
    return 301 https://$host$request_uri;  
}  
  
server {  
    listen 443 ssl default_server;  
    listen [::]:443 ssl default_server;  
    server_name DEINE_DOMAIN;  
  
    include snippets/letsencrypt-acme-challenge.conf;  
  
    ssl_certificate      /var/lib/dehydrated/certs/DEINE_DOMAIN/fullchain.pem;  
    ssl_certificate_key  /var/lib/dehydrated/certs/DEINE_DOMAIN/privkey.pem;  
    ssl_dhparam /etc/nginx/ssl/dhparam.pem;  
  
    ssl_session_cache shared:MozSSL:50m;  
    ssl_session_timeout 1d;  
    ssl_session_tickets off;  
    ssl_protocols TLSv1.2 TLSv1.3;  
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-  
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-  
POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-  
SHA384;  
    ssl_prefer_server_ciphers off;  
  
    server_tokens off;  
  
    add_header Strict-Transport-Security "max-age=15768000; includeSubDomains;"  
always;  
    add_header X-Frame-Options "SAMEORIGIN" always;  
    add_header X-Content-Type-Options "nosniff" always;  
    add_header Referrer-Policy "same-origin" always;  
    add_header Permissions-Policy "geolocation=(), camera=(), microphone=()" always;  
  
    client_max_body_size 256m;  
  
    location / {  
        proxy_pass          http://127.0.0.1:10200;  
        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;  
    }  
}
```

default_server und andere VHosts

Falls auf dem gleichen Nginx weitere VHosts mit Port 443 konfiguriert sind, muss sichergestellt werden dass nur ein VHost default_server für Port 443 ist. Andernfalls liefert Nginx beim SSL-Handshake möglicherweise das falsche Zertifikat.

Schritt — Konfiguration aktivieren

```
ln -s /etc/nginx/sites-available/DEINE_DOMAIN.conf /etc/nginx/sites-enabled/  
nginx -t  
systemctl reload nginx
```

nginx -t muss "syntax is ok" und "test is successful" ausgeben bevor du reload ausführst.

Prüfung

```
curl -s https://DEINE_DOMAIN/server/info/ | head -c 200
```

Erwartung: JSON-Antwort über HTTPS mit der Psono-Version.

Zertifikat prüfen:

```
echo | openssl s_client -connect DEINE_DOMAIN:443 -servername DEINE_DOMAIN  
2>/dev/null | openssl x509 -noout -subject
```

Erwartung: subject=CN=DEINE_DOMAIN

Phase 5 — Ersteinrichtung abschliessen

Schritt 5.1 — Ersten Benutzer registrieren

Öffne https://DEINE_DOMAIN im Browser und registriere den ersten Benutzer. Dieser wird später zum Administrator ernannt.

Schritt 5.2 — Administrator-Rechte vergeben

```
docker exec -ti psono-psono-combo-1 \  
python3 ./psono/manage.py promoteuser BENUTZERNAME@DEINE_DOMAIN superuser
```

Schritt 5.3 — Admin-Panel prüfen

Öffne https://DEINE_DOMAIN/portal/login und melde dich mit dem Admin-Benutzer an.

Schritt 5.4 — Automatische Backups einrichten

```
mkdir -p /opt/backups/psono
# crontab -e → folgende Zeilen hinzufügen:
0 2 * * * pg_dump -U psono -Ft psono > /opt/backups/psono/psono-$(date +%Y-%m-%d).tar
0 3 * * * find /opt/backups/psono -name "*.tar" -mtime +30 -delete
```



Backup der settings.yaml

Zusätzlich zur Datenbank sollte die `/opt/docker/psono/settings.yaml` regelmäßig gesichert werden — sie enthält die kryptographischen Schlüssel ohne die ein Datenbank-Backup wertlos ist.

Troubleshooting — Häufige Probleme

"Connection refused" beim curl-Test auf Port 10200

```
docker compose ps
docker compose logs psono-combo | tail -30
```

"no pg_hba.conf entry" in den Docker-Logs

Das Docker-Netzwerk hat eine unerwartete IP-Range. Prüfen welche IP der Container hat:

```
docker inspect psono-psono-combo-1 | grep IPAddress
```

Dann das passende Subnetz in `pg_hba.conf` eintragen und PostgreSQL neu laden:

```
systemctl reload postgresql
```

"could not connect" — PostgreSQL hört nicht auf Docker-IP

```
sudo -u postgres psql -c "SHOW listen_addresses;"
```

Falls nur localhost: ALTER SYSTEM SET listen_addresses aus Schritt 1.8 wiederholen und PostgreSQL neu starten.

"Redis ConnectionError" in den Docker-Logs

Cache ist noch aktiviert. `CACHE_ENABLE` in `settings.yaml` auf `FALSE` setzen und Container neu starten.

Nginx liefert falsches Zertifikat

```
grep -r "default_server" /etc/nginx/sites-enabled/
```

Nur ein VHost darf default_server für Port 443 sein. Alle anderen default_server-Direktiven für Port 443 entfernen.

"Invalid password" beim Psono-Login

Die kryptographischen Schlüssel in settings.yaml stimmen nicht mit der Datenbank überein. Die originale settings.yaml vom alten psono-combo Container verwenden — niemals neu generieren.

Dokument erstellt mit Claude · Anthropic · Juni 2026