# notify_push mit Docker Compose und Apache einrichten

## Zweck

Diese Anleitung beschreibt die Einrichtung der Nextcloud-App **Client Push** mit der technischen App-ID `notify_push` für mehrere getrennte Nextcloud-Docker-Projekte.

Die Anleitung ist für folgende Architektur ausgelegt:

- Nextcloud läuft als Docker-Compose-Projekt.
- Redis ist bereits vorhanden und in Nextcloud eingerichtet.
- Apache läuft auf dem Docker-Host als HTTPS-Reverse-Proxy.
- Jede Nextcloud-Instanz besitzt ein eigenes Docker-Netzwerk.
- Der Push-Dienst läuft als zusätzlicher Compose-Service.
- Port 7867 wird ausschließlich an `127.0.0.1` veröffentlicht.

`notify_push` reduziert regelmäßige Statusabfragen der Clients. Push-Nachrichten werden nach dem Best-Effort-Prinzip übertragen; die Clients führen weiterhin gelegentliche Prüfungen durch.

## 1. Platzhalter festlegen

| Platzhalter | Beispiel | Bedeutung |
|---|---|---|
| <PROJEKTVERZEICHNIS> | /home/docker/dein_projekt | Verzeichnis der compose Datei |
| <DOMAIN> | nextcloud.example.at | öffentliche Nextcloud Domain |
| <NEXTDCLOUD_IMAGE | nextcloud:34.0.2-apache | Image des Nextcloud Containers |
| <APP_SERVICE> | app | Compose Service der Nextcloud Anwendung |
| <APP_CONTAINERT> | nextcloud-dein_projekt | Name des Nextcloud Containers |
| <NETWORK> | dein_prjekt-netzwerk | Docker Netzwerk des Projektes |
| <PUSH_CONTAINER> | nextcloud-dein_projekt-notify_push | Name des push Containers |
| <APACHEE_BACKEND_PORT> | 8095 | Host-Port des Push Daemons |
| <PUSH_PORT> | 7867 | Port des Push Daemons |

Vor jeder Änderung:

```
cd <PROJEKTVERZEICHNIS>
cp docker-compose.yml docker-compose.yml.bak-$(date +%Y%m%d-%H%M%S)
```

## 2. Voraussetzungen prüfen

## 2.1 Redis prüfen

```
docker compose exec -u www-data <APP_SERVICE> php occ config:system:get
redis
```

zusätzlich

```
docker compose exec -u www-data <APP_SERVICE> php occ status
```

`notify_push` setzt einen funktionierenden Redis-Server voraus.

## 2.2. App installieren und aktivieren

```
docker compose exec -u www-data <APP_SERVICE> php occ app:install
notify_push
```

Falls die App bereits installiertist

```
docker compose exec -u www-data <APP_SERVICE> php occ app:enable notify_push
```

Status prüfen

```
docker compose exec -u www-data <APP_SERVICE> php occ app:list | grep -A3 -
B3 notify_push
```

## 2.3. Prozessorarchitektur und Binary prüfen

```

docker compose exec <APP_SERVICE> uname -m
```

Ausgabe bei einem AMD64/Intel Server

x86_64

Binary suchen

```
docker compose exec <APP_SERVICE> sh -c \
  'find /var/www/html/custom_apps/notify_push/bin \
  -type f -name notify_push -ls 2>/dev/null'
```

Für x86_64 muss diese Datei ausführbar sein:

```
/var/www/html/custom_apps/notify_push/bin/x86_64/notify_push
```

Falls erforderlich

```
chmod 755 custom_apps/notify_push/bin/x86_64/notify_push
```

# 3. Compose Service ergänzen

Der Push-Daemon wird als eigener Service im bestehenden Nextcloud-Projekt betrieben.

<blockquote>

Wichtig: Port 7867 darf nicht gleichzeitig beim Nextcloud-Service app und beim Dienst notify_push veröffentlicht werden.

</blockquote>==== 3.1. bei Bedarf Falsche Portfreigabe beim App-Service entfernen ====

Unter dem Nextcloud-Service darf nur der Web-Port stehen, zum Beispiel:

```
services:
  app:
    ports:
      - "127.0.0.1:<APACHE_BACKEND_PORT>:80"
```

## 3.2. Dienst mit notify_push

Den folgenden Service auf derselben Ebene wie app, db und redis ergänzen:

```
services:
  notify_push:
    image: <NEXTCLOUD_IMAGE>
    container_name: <PUSH_CONTAINER>
    restart: unless-stopped
    user: www-data

    entrypoint:
      - /var/www/html/custom_apps/notify_push/bin/x86_64/notify_push

    command:
      - /var/www/html/config/config.php

    environment:
      PORT: "7867"
      NEXTCLOUD_URL: "http://<APP_SERVICE>"
      TZ: "Europe/Vienna"

    depends_on:
      - <APP_SERVICE>
      - db
      - redis

    networks:
      - <NETWORK>

    ports:
```

```
      - "127.0.0.1:7867:7867"

    volumes:
      - nextcloud:/var/www/html:ro
      - ./config:/var/www/html/config:ro
      - ./custom_apps:/var/www/html/custom_apps:ro
      - /etc/localtime:/etc/localtime:ro

    secrets:
      - db_user_password
      - smtp_password
```

### 3.3. Volume Section anpassen

Die Volume-Namen und Bind-Mounts müssen zum vorhandenen app-Service passen.

Den App-Service anzeigen:

```
docker compose config | sed -n \
  '/^  <APP_SERVICE>:/,/^  [a-zA-Z0-9_-]*:/p'
```

Mindestens erforderlich sind normalerweise:

- Zugriff auf `/var/www/html/config`
- Zugriff auf `/var/www/html/custom_apps`
- Zugriff auf das Nextcloud-Volume, sofern der Binary-Pfad oder weitere Dateien daraus stammen
- dieselben Secrets, sofern `config.php` daraus Werte liest
- dasselbe Docker-Netzwerk wie Nextcloud, Datenbank und Redis

Die Mounts im Push-Service können schreibgeschützt (`:ro`) eingebunden werden.

## 4. Compose Konfiguration prüfen und starten

Syntax prüfen

```
cd <PROJEKTVERZEICHNIS>
docker compose config --quiet
```

Dienste anzeigen

```
docker compose config --services
```

`notify_push` muss in der Liste enthalten sein.

Container erstellen

```
docker compose up -d --force-recreate notify_push
```

Status prüfen

```
docker compose ps notify_push
```

erwartet wird

STATUS: Up
PORTS: 127.0.0.1:7867→7867/tcp

eventuell logs prüfen

```
docker compose logs --tail=100 notify_push
```

Bei `Restarting` oder `Exited`:

```
docker compose ps -a notify_push
docker compose logs --tail=200 notify_push
```

## 5. Apache Reverse-Proxy konfigurieren

Im HTTPS-VirtualHost der betreffenden Nextcloud-Domain müssen die Push-Regeln **vor** einer allgemeinen `ProxyPass / …`-Regel stehen.

Beispiel:

```
<VirtualHost *:443>
    ServerName <DOMAIN>

    ProxyPreserveHost On

    # Nextcloud Client Push
    ProxyPass        /push/ws  ws://127.0.0.1:7867/ws
    ProxyPass        /push/    http://127.0.0.1:7867/
    ProxyPassReverse /push/    http://127.0.0.1:7867/

    # Nextcloud
    ProxyPass        /  http://127.0.0.1:<APACHE_BACKEND_PORT>/
    ProxyPassReverse /  http://127.0.0.1:<APACHE_BACKEND_PORT>/

    # Bestehende TLS-, Header- und Zertifikatskonfiguration
    # bleibt unverändert.
</VirtualHost>
```

Die Reihenfolge ist wichtig!!

```
ProxyPass /push/ ...
ProxyPass / ...
```

### 5.1. Apache Module prüfen

```
sudo a2enmod proxy proxy_http proxy_wstunnel headers
```

Konfiguration prüfen

```
sudo apachectl configtest
```

bei erfolgreichem Test

```
sudo systemctl reload apache2
```

aktiven virtual-host prüfen

```
sudo systemctl reload apache2
```

Push Regeln suchen

```
sudo grep -RniE \
  'ServerName|ProxyPass.*push|ProxyPass[[:space:|]]+/' \
  /etc/apache2/sites-enabled /etc/apache2/conf-enabled
```

## 6. Docker-Netz als vertrauenswürdigen Proxy eintragen

Der Push-Container ruft Nextcloud intern über den Service-Namen auf:

```
NEXTCLOUD_URL: "http://<APP_SERVICE>"
```

Nextcloud muss daher:

1. das interne Docker-Netz als Proxy vertrauen und
2. den internen Service-Namen als Domain akzeptieren.

### 6.1 Docker-Netzwerk und Subnetz ermitteln

```
NETWORK=$(docker inspect <APP_CONTAINER> \
  --format '{{range $name, $conf := 
.NetworkSettings.Networks}}{{$name}}{{end}}')

SUBNET=$(docker network inspect "$NETWORK" \
  --format '{{range .IPAM.Config}}{{.Subnet}}{{end}}')

echo "Netzwerk: $NETWORK"
echo "Subnetz:  $SUBNET"
```

Beispiel:

Netzwerk: privat_privat-network
Subnetz: 172.29.0.0/16

### 6.2 Bestehende Proxy-Einträge prüfen

```
docker compose exec -u www-data <APP_SERVICE> php occ \
  config:system:get trusted_proxies
```

Beispiel:

127.0.0.1

### 6.3 Docker-Subnetz ergänzen

Den nächsten freien Index verwenden:

```
docker compose exec -u www-data <APP_SERVICE> php occ \
  config:system:set trusted_proxies 1 --value="$SUBNET"
```

Kontrolle

```
docker compose exec -u www-data <APP_SERVICE> php occ \
  config:system:get trusted_proxies
```

erwartet wird beispielsweise

127.0.0.1
172.29.0.0/16

**Nur kontrollierte Proxy-Adressen oder interne Docker-Netze eintragen. Vertrauenswürdige Proxys dürfen die von Nextcloud erkannte Client-IP beeinflussen.**

## 7. Internen Service-Namen als Trusted Domain eintragen

Vorhandene Einträge prüfen

```
docker compose exec -u www-data <APP_SERVICE> php occ \
  config:system:get trusted_domains
```

Den Service-Namen mit dem nächsten freien Index ergänzen:

```
docker compose exec -u www-data <APP_SERVICE> php occ \
  config:system:set trusted_domains 1 --value="<APP_SERVICE>"
```

Kontrolle

```
docker compose exec -u www-data <APP_SERVICE> php occ \
  config:system:get trusted_domains
```

# 8. Verbindung testen

## 8.1 Containerstatus

```
docker compose ps notify_push
docker compose logs --tail=100 notify_push
```

## 8.2 Lokaler Port

```
curl -i http://127.0.0.1:7867/test/cookie
```

eine Antwort wie

HTTP/1.1 400 Bad Request

ist bei einem direkten Aufruf über `127.0.0.1` möglich und zeigt bereits, dass der Push-Daemon antwortet.

`Connection refused` bedeutet dagegen:

- Container läuft nicht,
- Container startet ständig neu,
- Port ist nicht veröffentlicht oder
- kein Prozess lauscht im Container auf Port 7867.

## 8.3 Öffentlicher Pfad

```
curl -i https://<DOMAIN>/push/test/cookie
```

# 9. notify_push konfigurieren

Setup über den öffentlichen Push-Endpunkt ausführen:

```
docker compose exec -u www-data <APP_SERVICE> php occ \
  notify_push:setup https://<DOMAIN>/push
```

erwartetes Ergebnis

✓ redis is configured
✓ push server is receiving redis messages
✓ push server can load mount info from database

✓ push server can connect to the Nextcloud server
✓ push server is a trusted proxy
✓ push server is running the same version as the app
configuration saved

Gespeicherten Endpunkt prüfen:

```
docker compose exec -u www-data <APP_SERVICE> php occ \
  config:app:get notify_push base_endpoint
```

erwartet wird

https://<DOMAIN>/push

# 10. Firewall

Port 7867 wird nur an Loopback gebunden:

```
ports:
  - "127.0.0.1:7867:7867"
```

Daher ist keine öffentliche UFW-Freigabe nötig.

# 11. Client prüfen

Nach erfolgreichem Setup:

1. Nextcloud Desktop oder Nextcloud Talk Desktop vollständig abmelden.
2. Client auch im Infobereich beziehungsweise Tray beenden.
3. Client neu starten.
4. Konto erneut anmelden.
5. Diagnose prüfen.

Die Diagnose sollte nicht mehr melden:

```
notify_push app enabled = no
```

Metriken des Push-Dienstes prüfen:

```
docker compose exec -u www-data <APP_SERVICE> php occ \
  notify_push:metrics
```

# 12. Setup bei Neuinstallation notify_push möglicherweise

## durchführen

Bei Update folgendes beachten:

- `notify_push` zusammen mit der Nextcloud-App auf eine kompatible Version aktualisieren.
- Die feste IP-Adresse beibehalten.
- Die Einstellungen `trusted_proxies` nicht unbeabsichtigt überschreiben.
- Nach größeren Updates den Einrichtungstest erneut ausführen:

```
docker compose exec -T -u www-data app \
  php occ notify_push:setup \
  https://privat.waldhofer.at/push
```

From:
http://wiki.waldhofer.at/ - **Wiki von Franz**

Permanent link:
**http://wiki.waldhofer.at/doku.php?id=nextcloud:notify_push**

Last update: **2026/10/26 10:08**