Nextcloud in einer Proxmox VM und Docker

Warum das ganze?

Meine Intension

Ich betreibe seit langem Nextcloud als dedizierte VM im Proxmox ohne Docker. Es gibt schöne Anleitungen dazu, aber ich möchte das Aufsetzen eines solchen Systems verschlanken.

Aktuell sichere ich die ganze VM nachts ins NFS ohne inkrementelles Backup. Was über Nacht schon recht lange dauert und auch Fehleranfällig ist. Auch ist die Update-Strategie zwar automatisiert, aber dadurch habe ich wenig Kontrolle wenn das System nach dem Update abraucht.

Das führt im defekt Fall eine Wiederherstellung zu einem enormen Arbeitsaufwand. Und die Familie ist nicht happy, wenn ich sie zu einem System überrede, was dann nicht funktioniert.

Und nicht zuletzt belege ich unnötigen Speicherplatz, weil es immer ein Vollbackup ist.

Außerdem wollte ich für mich eine Anleitung.

Wer hat mir geholfen

Viele Impulse und auch Lösungen stammen aus dem Internet und von ChatGPT. Behaltet das im Hinterkopf!

Voraussetzungen

- Du hast eine VM im Proxmox angelegt, die entsprechend RAM, CPU und Festplattenspeicher hat
- dein Linux läuft und Du kannst Dich per SSH auf die Maschine einwählen
- ich halte alle Daten (www, DB und Files) alle in der VM, es wurde für den Betrieb kein externes Filesystem (z.B. NFS angebunden)
- ich verwende nur NFS aus einer Synology für das Backup
- Du solltest über einen vernünftigen Uplink zum Internet verfügen. In meinem Fall 100 Mbit/s Uplink-Verbindung

Was soll die Nextcloud können?

Ich möchte in der Nextcloud folgende Feature haben

- **Files** inklusive selbst gehosteten Collabora Online Server, PDF Unterstützung, Notizen usw..
- **Talk** als **Messanger** und **Besprechungstool** ohne Hochleistungsbackend (ich habe max. 3 gleichzeitige Teilnehmer in Videokonferenzen, es soll auch ein Ersatz für Teams oder WhatsApp in der Familie sein.
- **Kontakte** und **Kalender** auch für die Synchronisation mit einem Smartphone oder Tablet (in meinem Fall von Apple). Dadurch können auch Kontaktdaten in der Familie geteilt werden.
- **Zugriff aus dem Internet** über einen dynamik DNS-Anbieter **nur über https** (LetsEncrypt) mit automatischer Zertifikatsaktualisierung
- **zwei Faktor-Authentifizierung** für die Webseite und **App-Passwort** für Apps auf den Smartphone

Was diese Anleitung nicht macht!

- Sie nimmt Dir nicht ab selbst zu denken und alles geschriebene zu prüfen.
- Du bist eigenverantwortlich für das was Du tust.
- Denke vorher nach und tippe danach!

Die Docker Installation

Docker installieren

Geht auf Debian | Docker Docs und führt bitte die beschriebene Anleitung aus!

Damit habt Ihr nicht die Paketquellen für docker von Debian, sondern direkt von docker.com, die deutlich aktueller sind.

Danach sollte Docker verfügbar sein und mit diesem Kommando als root könnt Ihr das prüfen:

```
docker ps
```

sudo installieren

In deiner Debianinstallation wird beim Installationsprozess ein zweiter User mit angelegt. Diesen verwenden wir als Verwalter für die Dockercontainer in meinem Fall ist es der **dockeruser**.

Wir loggen uns als root auf dem System per ssh ein.

Nun führen wir folgende Kommandos aus:

```
apt update && apt upgrade -y && apt install sudo
```

Anschließend weisen wir dem dockeruser die Gruppe docker zu, damit er dockerkommandos ausführen kann.

Zuweisung des dockerusers zur Gruppe docker

Prüfen ob die Gruppe `docker` existiert

```
getent group docker
```

Falls nicht (diese sollte aber mit der Dockerinstallation schon vorhanden sein):

```
sudo groupadd docker
```

dockeruser zur Docker-Gruppe hinzufügen:

```
usermod -aG docker dockeruser
```

Erster Versuch als dedizierter Dockerverwalter

Wir wechseln zu unserem **dockeruser** und in das Homeverzeichnis des dockeruser:

```
su dockeruser
cd ~
```

Anschließend prüfen wir ob wir dockerkommandos ausführen dürfen:

```
sudo docker ps
```

Die Nextcloud docker-compose.yml

Nun kann es endlich mit Netxcloud losgehen

Dieses Template wird immer mal aktualisiert und ist ausschließlich für eine Neuinstallation gedacht!

Der Updateprozess wird später erklärt!

Anlegen des Programmordners

Ich habe folgende Programmstruktur bei mir angelegt: **/home/dockeruser/docker/nextcloud**

Dort werden alle benötigten Dateien zum Betrieb der Nextcloud abgelegt.

Alle nachfolgenden Schritte in dieser Anleitung beziehen sich immer auf diese Struktur!

Folgende Kommandos fügt Ihr nun als **dockeruser** aus:

```
mkdir ~/docker && mkdir ~/docker/nextcloud && cd ~/docker/nextcloud
```

Nun solltest ich im folgenden Verzeichnis sein:

```
/home/dockeruser/docker/nextcloud#
```

Erstellen der Datei docker-compose.yml

Mit folgendem Kommando öffnet Ihr den Editor für die docker-compose.yml:

```
nano docker-compose.yml
```

Nun wird folgender Code dort hinein kopiert.

```
services:
  db:
    image: postgres:16
    container_name: nextcloud-db
    restart: unless-stopped
    environment:
      POSTGRES_DB: nextcloud
      POSTGRES_USER: nextcloud
      POSTGRES_PASSWORD: EIN_SICHERES_PASSWORT_FESTLEGEN
    volumes:
      - /home/dockeruser/docker/nextcloud/db:/var/lib/postgresql/data

  redis:
    image: redis:7-alpine
    container_name: nextcloud-redis
    restart: unless-stopped
    command: ["redis-server", "--appendonly", "yes"]
    volumes:
      - /home/dockeruser/docker/nextcloud/redis:/data

  app:
    image: nextcloud:31-fpm
    container_name: nextcloud-app
    restart: unless-stopped
    depends_on:
      - db
      - redis
    environment:
      POSTGRES_HOST: db
      POSTGRES_DB: nextcloud
      POSTGRES_USER: nextcloud
      POSTGRES_PASSWORD: EIN_SICHERES_PASSWORT_FESTLEGEN
      REDIS_HOST: redis
    volumes:
      -  /home/dockeruser/docker/nextcloud/nextcloud:/var/www/html
      -  /home/dockeruser/docker/nextcloud/config:/var/www/html/config

  web:
    image: nginx:alpine
    container_name: nextcloud-web
```

```
    restart: unless-stopped
    depends_on:
      - app
    ports:
      - "8088:80"
    volumes:
      -  /home/dockeruser/docker/nextcloud/nextcloud:/var/www/html:ro
      -  /home/dockeruser/docker/nextcloud/config:/var/www/html/config:ro
      -
/home/dockeruser/docker/nextcloud/nginx/default.conf:/etc/nginx/conf.d/default.conf:ro

  collabora:
    image: collabora/code:latest
    container_name: collabora
    restart: unless-stopped
    environment:
      - domain=office\\.mein\\.dyndns\\.local
      - username=EINEN_BENUTZERNAMEN_FESTLEGEN
      - password=EIN_SICHERES_PASSWORT_FESTLEGEN
      - extra_params=--o:ssl.enable=false --o:ssl.termination=true
    ports:
      - "9980:9980"

  cron:
    image: nextcloud:31-fpm
    container_name: nextcloud-cron
    restart: unless-stopped
    depends_on:
      - db
      - redis
    volumes:
      -  /home/dockeruser/docker/nextcloud/nextcloud:/var/www/html
      -  /home/dockeruser/docker/nextcloud/config:/var/www/html/config
    entrypoint: /cron.sh
```

WICHTIG: Handelt hier vorsichtig und mit wirklich starken Passwörtern!

Im nano ändert Ihr vor den Speichern den Inhalt der Variablen folgender Zeilen **9, 32, 56, 58, 59 !**

Der Reverse-Proxy (traefik)

Um die Seite im Internet bereit zu stellen benutzen wir einen Reverse-Proxy der den Netxtcloud-Dienst ausschließlich über TLS-Verschlüsselung anbietet (https). Im Heimsetzerk ist die Nextcloud über http erreichbar, wird aber darüber nicht genutzt.

Ich betreibe einen Traefik Dienst im Docker der als ein HTTP-Reverse-Proxy und Load Balancer dient, der die Bereitstellung von Microservices erleichtert. Dieser läuft auf einer anderen Maschine, da ich dort viele Microservices laufen lasse (unter anderen dieses Wiki) und ich will die Nextcloud-Instanz nicht auch dort noch laufen lassen, aus Performanz gründen.

Zeile 56 dient zur Definition **FQDNS** für Collabora. Die \\. müssen drin bleiben!

Die initiale nginx default.conf

Bitte führe folgenden Befehl in der Konsole aus:

```
cat > /home/dockeruser/docker/nextcloud/nginx/default.conf <<'EOF'
upstream php-handler {
  server app:9000;
}

server {
  listen 80;
  server_name _;

  include /etc/nginx/mime.types;

  types { text/javascript mjs; }

  client_max_body_size 10G;
  fastcgi_buffers 64 4K;

  root /var/www/html;

  add_header Referrer-Policy "no-referrer" always;
  add_header X-Content-Type-Options "nosniff" always;
  add_header X-Frame-Options "SAMEORIGIN" always;
  add_header X-Permitted-Cross-Domain-Policies "none" always;
  add_header X-Robots-Tag "noindex, nofollow" always;
  add_header X-XSS-Protection "1; mode=block" always;
```

```
  add_header Strict-Transport-Security "max-age=15552000" always;

  index index.php index.html /index.php$request_uri;

  location = /robots.txt {
    allow all;
    log_not_found off;
    access_log off;
  }

  location = /.well-known/carddav { return 301 /remote.php/dav/; }
  location = /.well-known/caldav  { return 301 /remote.php/dav/; }

  location / {
    try_files $uri $uri/ /index.php$request_uri;
  }

  location ~ \.php(?:$|/) {
    rewrite
^/(?!index|remote|public|cron|core\/ajax\/update|status|ocs\/v[12]|updater\/.+|ocs-
provider\/.+|.+\/richdocumentscode\/proxy) /index.php$request_uri;

    fastcgi_split_path_info ^(.+?\.php)(/.*)$;
    set $path_info $fastcgi_path_info;

    try_files $fastcgi_script_name =404;

    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param PATH_INFO $path_info;
    fastcgi_param HTTPS on;
    fastcgi_param modHeadersAvailable true;
    fastcgi_param front_controller_active true;

    fastcgi_pass php-handler;
    fastcgi_intercept_errors on;
    fastcgi_request_buffering off;
  }

  location ~ \.(?:css|js|mjs|svg|gif|png|jpg|ico|wasm|tflite|map)$ {
```

```
    try_files $uri /index.php$request_uri;
    add_header Cache-Control "public, max-age=15778463, immutable";
    access_log off;
    expires 6M;
  }

  location ~ \.(?:woff2?|otf|ttf)$ {
    try_files $uri /index.php$request_uri;
    expires 7d;
    access_log off;
  }

  location /remote {
    return 301 /remote.php$request_uri;
  }

  location /updater {
    try_files $uri/ =404;
    index index.php;
  }
}
EOF
```

Backup Scripte

Intension

Ich möchte eine automatisierte Datensicherung auf ein NFS, was auf einer Synology läuft. ChatGPT hat mir Restic vorgeschlagen und auch begründet:

Restic Backups sind immer verschlüsselt (AES-256), inkrementell und backend unabhängig. In meinen Fall habe ich max. 300 GB im Backup zu verwalten. Restic legt Snapshots statt „Dateikopien“ an. Jeder Snapshot ist ein vollständiger Zustand aber physisch werden nur Änderungen gespeichert.

Wenn du z.B. eine 10 GB Datei hast und sich 1 MB ändert:

- rsync: kopiert neu
- tar: kopiert neu
- Restic: speichert nur geänderte Blöcke

Für Nextcloud mit vielen Versionen und Office-Dateien: sehr gut und Restic ist einfacher in Docker/NAS/Cloud Setups

Wir bauen mit folgenden Skripten folgendes Szenario:

- automatisches Backup
- inkrementell
- verschlüsselt
- Retention
- Integrity Check
- Restore-Test
- Health Report
- Lock Handling
- Monitoring
- CSV Logging

Daher nutzen wir Restic.

```
nano /root/.nextcloud-backup.env
```

Ändere bitte alle Einträge mit Deinen Werten!

```
#!/bin/bash
set -euo pipefail

# =========================
# Nextcloud Restic Backup
# - Single snapshot per run (files + DB dump)
# - NFS/NAS target wait (mounted + writable)
# - Retention + prune (warning-only on failure/lock)
# - Integrity check (subset)
# - CSV duration logging
# - Pushover OK/FAIL (+ warning on retention failure)
# =========================

# ---- Load config/secrets ----
ENV_FILE="/root/.nextcloud-backup.env"
if [[ -f "$ENV_FILE" ]]; then
  # shellcheck disable=SC1090
  source "$ENV_FILE"
else
  echo "ERROR: Env file not found: $ENV_FILE"
  exit 1
fi

# ---- Required env vars ----
: "${RESTIC_REPOSITORY:?RESTIC_REPOSITORY not set}"
: "${RESTIC_PASSWORD_FILE:?RESTIC_PASSWORD_FILE not set}"
: "${BACKUP_MOUNT:?BACKUP_MOUNT not set}"

: "${NC_BASE:?NC_BASE not set}"
: "${NC_CONFIG_PATHS:?NC_CONFIG_PATHS not set}"
: "${NC_DATA_PATH:?NC_DATA_PATH not set}"
: "${NC_COMPOSE_FILE:?NC_COMPOSE_FILE not set}"
: "${NC_META_DIR:?NC_META_DIR not set}"

: "${NC_APP_CONTAINER:?NC_APP_CONTAINER not set}"
: "${NC_DB_CONTAINER:?NC_DB_CONTAINER not set}"

: "${PGUSER:?PGUSER not set}"
: "${PGDB:?PGDB not set}"
```

```
: "${KEEP_DAILY:?KEEP_DAILY not set}"
: "${KEEP_WEEKLY:?KEEP_WEEKLY not set}"
: "${KEEP_MONTHLY:?KEEP_MONTHLY not set}"
: "${READ_SUBSET:?READ_SUBSET not set}"

: "${PUSHOVER_TOKEN:?PUSHOVER_TOKEN not set}"
: "${PUSHOVER_USER:?PUSHOVER_USER not set}"
# PUSHOVER_DEVICE is optional

export RESTIC_REPOSITORY
export RESTIC_PASSWORD_FILE

# ---- Logging ----
LOGFILE="/var/log/nextcloud-backup.log"
CSVLOG="/var/log/nextcloud-backup.csv"
HOSTNAME="$(hostname -s)"

exec >> "$LOGFILE" 2>&1

log() { echo "[$(date '+%F %T')] $*"; }

csv_log() {
  local STATUS="$1"     # OK / FAIL
  local DURATION="$2"   # seconds
  local SNAPSHOT="${3:-}"
  local TS
  TS="$(date --iso-8601=seconds)"

  if [[ ! -f "$CSVLOG" ]]; then
    echo "timestamp,host,status,duration_seconds,snapshot_id" > "$CSVLOG"
    chmod 600 "$CSVLOG" || true
  fi

  echo "${TS},${HOSTNAME},${STATUS},${DURATION},${SNAPSHOT}" >> "$CSVLOG"
}

# ---- Pushover ----
pushover() {
  local TITLE="$1"
  local MESSAGE="$2"
```

```
  local PRIORITY="${3:-0}"

  local DEVICE_ARG=()
  if [[ -n "${PUSHOVER_DEVICE:-}" ]]; then
    DEVICE_ARG=(-F "device=$PUSHOVER_DEVICE")
  fi

  curl -s \
    -F "token=$PUSHOVER_TOKEN" \
    -F "user=$PUSHOVER_USER" \
    "${DEVICE_ARG[@]}" \
    -F "title=$TITLE" \
    -F "priority=$PRIORITY" \
    -F "message=$MESSAGE" \
    https://api.pushover.net/1/messages.json >/dev/null || true
}

# ---- Wait for backup target (mounted + writable) ----
wait_for_backup_target() {
  local TARGET_MOUNT="$BACKUP_MOUNT"
  local MAX_WAIT=900        # 15 minutes
  local INTERVAL=15         # seconds
  local waited=0

  log "Checking backup target: ${TARGET_MOUNT}"

  while true; do
    if mountpoint -q "$TARGET_MOUNT"; then
      if timeout 5 sh -c "echo test > '${TARGET_MOUNT}/.nc_backup_writetest'"; then
        rm -f "${TARGET_MOUNT}/.nc_backup_writetest" >/dev/null 2>&1 || true
        log "Backup target OK (mounted + writable)"
        return 0
      fi
      log "Backup target mounted but not writable yet."
    else
      log "Backup target not mounted yet."
    fi

    if (( waited >= MAX_WAIT )); then
      log "ERROR: Backup target not ready after ${MAX_WAIT}s"
```

```
      pushover "Nextcloud Backup FAILED" "Backup target (${TARGET_MOUNT}) not
available/writable after ${MAX_WAIT}s. Aborting." 1
      exit 1
    fi

    sleep "$INTERVAL"
    waited=$((waited + INTERVAL))
  done
}

# ---- Maintenance cleanup ----
cleanup() {
  docker exec -u www-data "$NC_APP_CONTAINER" php occ maintenance:mode --off >/dev/null 2>&1
|| true
}

# ---- Local lock to prevent concurrent jobs (backup vs restore-test) ----
LOCKFILE="/var/lock/nextcloud-restic.lock"
exec 9>"$LOCKFILE"
flock -n 9 || { echo "Another backup/restore job is running. Exiting."; exit 0; }

START_TS=$(date +%s)
SNAPSHOT_ID=""

on_error() {
  local exit_code=$?
  local now dur
  now=$(date +%s)
  dur=$((now-START_TS))

  log "ERROR: Backup failed (exit=${exit_code}) after ${dur}s"
  csv_log "FAIL" "$dur" "${SNAPSHOT_ID:-}"
  pushover "Nextcloud Backup FAILED" "Backup failed after ${dur}s (exit=${exit_code}). Check
${LOGFILE}" 1

  cleanup
  exit "$exit_code"
}

trap on_error ERR
```

```
trap cleanup EXIT

log "========== START BACKUP =========="

# Wait for NAS/NFS target before doing anything
wait_for_backup_target

log "Enabling maintenance mode"
docker exec -u www-data "$NC_APP_CONTAINER" php occ maintenance:mode --on

# ---- Create DB dump file locally so it is included in the SAME restic snapshot ----
META_PATH="$NC_BASE/$NC_META_DIR"
DB_DUMP_FILE="$META_PATH/db.sql"

mkdir -p "$META_PATH"
chmod 700 "$META_PATH" || true

log "Creating database dump file: $DB_DUMP_FILE"
docker exec -t "$NC_DB_CONTAINER" pg_dump -U "$PGUSER" -d "$PGDB" > "$DB_DUMP_FILE"
chmod 600 "$DB_DUMP_FILE" || true

# ---- Build backup path list from env ----
BACKUP_PATHS=()

# Config directories (only include existing ones)
for cfg in $NC_CONFIG_PATHS; do
  FULL_CFG="$NC_BASE/$cfg"
  if [[ -d "$FULL_CFG" ]]; then
    BACKUP_PATHS+=("$FULL_CFG")
  fi
done

# Compose file
if [[ -f "$NC_BASE/$NC_COMPOSE_FILE" ]]; then
  BACKUP_PATHS+=("$NC_BASE/$NC_COMPOSE_FILE")
else
  log "WARNING: Compose file not found: $NC_BASE/$NC_COMPOSE_FILE"
fi

# Data directory
```

```
if [[ -d "$NC_BASE/$NC_DATA_PATH" ]]; then
  BACKUP_PATHS+=("$NC_BASE/$NC_DATA_PATH")
else
  log "WARNING: Data directory not found: $NC_BASE/$NC_DATA_PATH"
fi

# Meta directory (db dump etc.)
BACKUP_PATHS+=("$META_PATH")

if [[ ${#BACKUP_PATHS[@]} -eq 0 ]]; then
  log "ERROR: No valid backup paths found. Check env: NC_BASE / NC_CONFIG_PATHS / NC_DATA_PATH
/ NC_COMPOSE_FILE / NC_META_DIR."
  exit 1
fi

log "Restic backup paths: ${BACKUP_PATHS[*]}"
log "Running restic backup (single snapshot: files + db dump)"

# Do not back up Redis data (cache only)
restic backup \
  --tag nextcloud --tag daily \
  --exclude "$NC_BASE/redis" \
  --exclude "$NC_BASE/redis/**" \
  "${BACKUP_PATHS[@]}"

log "Removing local DB dump file (kept inside restic snapshot)"
rm -f "$DB_DUMP_FILE" || true

log "Disabling maintenance mode"
docker exec -u www-data "$NC_APP_CONTAINER" php occ maintenance:mode --off

# Snapshot ID (best-effort)
SNAPSHOT_ID="$(restic snapshots --tag nextcloud --latest 1 --json 2>/dev/null \
  | grep -m1 '"short_id"' \
  | sed -E 's/.*"short_id": ?"([^"]+)".*/\1/' || true)"

# ---- Retention (warning-only if it fails, do not fail the entire backup) ----
log "Retention: forget/prune (daily=${KEEP_DAILY}, weekly=${KEEP_WEEKLY},
monthly=${KEEP_MONTHLY})"
if ! restic forget \
```

```
  --tag nextcloud \
  --keep-daily "$KEEP_DAILY" \
  --keep-weekly "$KEEP_WEEKLY" \
  --keep-monthly "$KEEP_MONTHLY" \
  --prune; then

  log "WARNING: Retention/prune failed (backup itself is OK)."
  if restic list locks >/dev/null 2>&1; then
    log "WARNING: Repository appears to be locked. Retention skipped."
    pushover "Nextcloud Backup WARNING" \
      "Restic repository is locked. Retention/prune was skipped. Run 'restic unlock' if the
lock is stale." 1
  else
    pushover "Nextcloud Backup WARNING" \
      "Retention/prune failed (not a repo lock). Check ${LOGFILE}." 1
  fi
fi

log "Integrity check (read-data-subset=${READ_SUBSET})"
restic check --read-data-subset="$READ_SUBSET" >/dev/null

END_TS=$(date +%s)
DUR=$((END_TS-START_TS))

log "Backup finished in ${DUR}s (snapshot=${SNAPSHOT_ID:-unknown})"
log "========== END BACKUP =========="

csv_log "OK" "$DUR" "${SNAPSHOT_ID:-}"
pushover "Nextcloud Backup OK" "Backup successful in ${DUR}s (snapshot=${SNAPSHOT_ID:-
unknown})."
```

```
sudo chown root:root /root/.nextcloud-backup.env
sudo chmod 600 /root/.nextcloud-backup.env
```

Datei: /root/nextcloud-backup.sh

```
#!/bin/bash
set -euo pipefail

# ========================
```

```
# Nextcloud Restic Backup
# ========================

# ---- Load config/secrets ----
ENV_FILE="/root/.nextcloud-backup.env"
if [[ -f "$ENV_FILE" ]]; then
  source "$ENV_FILE"
else
  echo "ERROR: Env file not found: $ENV_FILE"
  exit 1
fi

# ---- Required env vars ----
: "${RESTIC_REPOSITORY:?RESTIC_REPOSITORY not set}"
: "${RESTIC_PASSWORD_FILE:?RESTIC_PASSWORD_FILE not set}"
: "${BACKUP_MOUNT:?BACKUP_MOUNT not set}"

: "${NC_BASE:?NC_BASE not set}"
: "${NC_CONFIG_PATHS:?NC_CONFIG_PATHS not set}"
: "${NC_DATA_PATH:?NC_DATA_PATH not set}"
: "${NC_COMPOSE_FILE:?NC_COMPOSE_FILE not set}"

: "${NC_APP_CONTAINER:?NC_APP_CONTAINER not set}"
: "${NC_DB_CONTAINER:?NC_DB_CONTAINER not set}"

: "${PGUSER:?PGUSER not set}"
: "${PGDB:?PGDB not set}"

: "${KEEP_DAILY:?KEEP_DAILY not set}"
: "${KEEP_WEEKLY:?KEEP_WEEKLY not set}"
: "${KEEP_MONTHLY:?KEEP_MONTHLY not set}"
: "${READ_SUBSET:?READ_SUBSET not set}"

: "${PUSHOVER_TOKEN:?PUSHOVER_TOKEN not set}"
: "${PUSHOVER_USER:?PUSHOVER_USER not set}"
# PUSHOVER_DEVICE is optional

export RESTIC_REPOSITORY
export RESTIC_PASSWORD_FILE
```

```
# ---- Logging ----
LOGFILE="/var/log/nextcloud-backup.log"
CSVLOG="/var/log/nextcloud-backup.csv"
HOSTNAME="$(hostname -s)"

exec >> "$LOGFILE" 2>&1

log() { echo "[$(date '+%F %T')] $*"; }

csv_log() {
  local STATUS="$1"     # OK / FAIL
  local DURATION="$2"   # seconds
  local SNAPSHOT="${3:-}"
  local TS
  TS="$(date --iso-8601=seconds)"

  if [[ ! -f "$CSVLOG" ]]; then
    echo "timestamp,host,status,duration_seconds,snapshot_id" > "$CSVLOG"
    chmod 600 "$CSVLOG" || true
  fi

  echo "${TS},${HOSTNAME},${STATUS},${DURATION},${SNAPSHOT}" >> "$CSVLOG"
}

# ---- Pushover ----
pushover() {
  local TITLE="$1"
  local MESSAGE="$2"
  local PRIORITY="${3:-0}"

  local DEVICE_ARG=()
  if [[ -n "${PUSHOVER_DEVICE:-}" ]]; then
    DEVICE_ARG=(-F "device=$PUSHOVER_DEVICE")
  fi

  curl -s \
    -F "token=$PUSHOVER_TOKEN" \
    -F "user=$PUSHOVER_USER" \
    "${DEVICE_ARG[@]}" \
    -F "title=$TITLE" \
```

```
    -F "priority=$PRIORITY" \
    -F "message=$MESSAGE" \
    https://api.pushover.net/1/messages.json >/dev/null || true
}

# ---- Wait for backup target (mounted + writable) ----
wait_for_backup_target() {
  local TARGET_MOUNT="$BACKUP_MOUNT"
  local MAX_WAIT=900        # 15 minutes
  local INTERVAL=15         # seconds
  local waited=0

  log "Checking backup target: ${TARGET_MOUNT}"

  while true; do
    if mountpoint -q "$TARGET_MOUNT"; then
      if timeout 5 sh -c "echo test > '${TARGET_MOUNT}/.nc_backup_writetest'"; then
        rm -f "${TARGET_MOUNT}/.nc_backup_writetest" >/dev/null 2>&1 || true
        log "Backup target OK (mounted + writable)"
        return 0
      fi
      log "Backup target mounted but not writable yet."
    else
      log "Backup target not mounted yet."
    fi

    if (( waited >= MAX_WAIT )); then
      log "ERROR: Backup target not ready after ${MAX_WAIT}s"
      pushover "Nextcloud Backup FAILED" "Backup target (${TARGET_MOUNT}) not
available/writable after ${MAX_WAIT}s. Aborting." 1
      exit 1
    fi

    sleep "$INTERVAL"
    waited=$((waited + INTERVAL))
  done
}

# ---- Maintenance cleanup ----
cleanup() {
```

```
  docker exec -u www-data "$NC_APP_CONTAINER" php occ maintenance:mode --off >/dev/null 2>&1
|| true
}

START_TS=$(date +%s)
SNAPSHOT_ID=""

on_error() {
  local exit_code=$?
  local now dur
  now=$(date +%s)
  dur=$((now-START_TS))

  log "ERROR: Backup failed (exit=${exit_code}) after ${dur}s"
  csv_log "FAIL" "$dur" "${SNAPSHOT_ID:-}"
  pushover "Nextcloud Backup FAILED" "Backup failed after ${dur}s (exit=${exit_code}). Check
${LOGFILE}" 1

  cleanup
  exit "$exit_code"
}

trap on_error ERR
trap cleanup EXIT

log "========== START BACKUP =========="

# Wait for NAS/NFS target before doing anything
wait_for_backup_target

log "Enabling maintenance mode"
docker exec -u www-data "$NC_APP_CONTAINER" php occ maintenance:mode --on

# ---- Build backup path list from env ----
BACKUP_PATHS=()

# Config directories (only include existing ones)
for cfg in $NC_CONFIG_PATHS; do
  FULL_CFG="$NC_BASE/$cfg"
  if [[ -d "$FULL_CFG" ]]; then
```

```
    BACKUP_PATHS+=("$FULL_CFG")
  fi
done

# Compose file
if [[ -f "$NC_BASE/$NC_COMPOSE_FILE" ]]; then
  BACKUP_PATHS+=("$NC_BASE/$NC_COMPOSE_FILE")
else
  log "WARNING: Compose file not found: $NC_BASE/$NC_COMPOSE_FILE"
fi

# Data directory
if [[ -d "$NC_BASE/$NC_DATA_PATH" ]]; then
  BACKUP_PATHS+=("$NC_BASE/$NC_DATA_PATH")
else
  log "WARNING: Data directory not found: $NC_BASE/$NC_DATA_PATH"
fi

if [[ ${#BACKUP_PATHS[@]} -eq 0 ]]; then
  log "ERROR: No valid backup paths found. Check NC_BASE / NC_CONFIG_PATHS / NC_DATA_PATH /
NC_COMPOSE_FILE."
  exit 1
fi

log "Restic backup paths: ${BACKUP_PATHS[*]}"
log "Running restic backup (files)"
restic backup --tag nextcloud --tag daily "${BACKUP_PATHS[@]}"

log "Running restic backup (database dump via stdin -> db.sql)"
docker exec -t "$NC_DB_CONTAINER" pg_dump -U "$PGUSER" -d "$PGDB" \
  | restic backup --stdin --stdin-filename "db.sql" --tag nextcloud --tag daily

log "Disabling maintenance mode"
docker exec -u www-data "$NC_APP_CONTAINER" php occ maintenance:mode --off

# Snapshot ID (best-effort)
SNAPSHOT_ID="$(restic snapshots --tag nextcloud --latest 1 --json 2>/dev/null \
  | grep -m1 '"short_id"' \
  | sed -E 's/.*"short_id": ?"([^"]+)".*/\1/' || true)"
```

```
log "Retention: forget/prune (daily=${KEEP_DAILY}, weekly=${KEEP_WEEKLY},
monthly=${KEEP_MONTHLY})"
restic forget \
  --tag nextcloud \
  --keep-daily "$KEEP_DAILY" \
  --keep-weekly "$KEEP_WEEKLY" \
  --keep-monthly "$KEEP_MONTHLY" \
  --prune

log "Integrity check (read-data-subset=${READ_SUBSET})"
restic check --read-data-subset="$READ_SUBSET" >/dev/null

END_TS=$(date +%s)
DUR=$((END_TS-START_TS))

log "Backup finished in ${DUR}s (snapshot=${SNAPSHOT_ID:-unknown})"
log "========== END BACKUP =========="

csv_log "OK" "$DUR" "${SNAPSHOT_ID:-}"
pushover "Nextcloud Backup OK" "Backup successful in ${DUR}s (snapshot=${SNAPSHOT_ID:-
unknown})."
```

```
sudo chown root:root /root/nextcloud-backup.sh
sudo chmod 700 /root/nextcloud-backup.sh
```

Datei: /root/nextcloud-restore-test.sh

```
#!/bin/bash
set -euo pipefail

# ========================
# Monthly Restic Restore Test
# ========================

ENV_FILE="/root/.nextcloud-backup.env"
if [[ -f "$ENV_FILE" ]]; then
  source "$ENV_FILE"
else
  echo "ERROR: Env file not found: $ENV_FILE"
  exit 1
```

```
fi

: "${RESTIC_REPOSITORY:?RESTIC_REPOSITORY not set}"
: "${RESTIC_PASSWORD_FILE:?RESTIC_PASSWORD_FILE not set}"
: "${PUSHOVER_TOKEN:?PUSHOVER_TOKEN not set}"
: "${PUSHOVER_USER:?PUSHOVER_USER not set}"

export RESTIC_REPOSITORY
export RESTIC_PASSWORD_FILE

LOGFILE="/var/log/nextcloud-restore-test.log"
exec >> "$LOGFILE" 2>&1

log() { echo "[$(date '+%F %T')] $*"; }

pushover() {
  local TITLE="$1"
  local MESSAGE="$2"
  local PRIORITY="${3:-0}"

  local DEVICE_ARG=()
  if [[ -n "${PUSHOVER_DEVICE:-}" ]]; then
    DEVICE_ARG=(-F "device=$PUSHOVER_DEVICE")
  fi

  curl -s \
    -F "token=$PUSHOVER_TOKEN" \
    -F "user=$PUSHOVER_USER" \
    "${DEVICE_ARG[@]}" \
    -F "title=$TITLE" \
    -F "priority=$PRIORITY" \
    -F "message=$MESSAGE" \
    https://api.pushover.net/1/messages.json >/dev/null || true
}

# Helper: pick first real path from `restic find` output
pick_find_path() {
  # Prefer lines that start with "/" (real paths)
  local p
  p="$(restic find latest "$1" 2>/dev/null | grep '^/' | head -n 1 || true)"
```

```
  if [[ -n "${p:-}" ]]; then
    echo "$p"
    return 0
  fi
  # Fallback: ignore informational lines, take first remaining line
  p="$(restic find latest "$1" 2>/dev/null \
      | grep -v '^Found matching entries' \
      | grep -v '^repository ' \
      | head -n 1 || true)"
  echo "$p"
}

TARGET="/tmp/restore-test-monthly"
rm -rf "$TARGET"
mkdir -p "$TARGET"

log "========== MONTHLY RESTORE TEST =========="
log "Repo: $RESTIC_REPOSITORY"

SNAPSHOT_ID="$(restic snapshots --latest 1 --json 2>/dev/null \
  | grep -m1 '"short_id"' \
  | sed -E 's/.*"short_id": ?"([^"]+)".*/\1/' || true)"

if [[ -z "${SNAPSHOT_ID:-}" ]]; then
  log "ERROR: No snapshots found"
  pushover "Nextcloud Restore-Test FAILED" "No restic snapshots found." 1
  exit 1
fi

log "Latest snapshot: $SNAPSHOT_ID"

CFG_PATH="$(pick_find_path "config.php")"
DB_PATH="$(pick_find_path "db.sql")"

if [[ -z "${CFG_PATH:-}" ]]; then
  log "ERROR: config.php not found in latest snapshot"
  pushover "Nextcloud Restore-Test FAILED" "config.php not found in latest snapshot." 1
  exit 1
fi
```

```
if [[ -z "${DB_PATH:-}" ]]; then
  log "ERROR: db.sql not found in latest snapshot"
  pushover "Nextcloud Restore-Test FAILED" "db.sql not found in latest snapshot." 1
  exit 1
fi

# Use relative paths for restore (more compatible across restic versions)
CFG_REL="${CFG_PATH#/}"
DB_REL="${DB_PATH#/}"

log "Restoring config: $CFG_REL"
restic restore latest --target "$TARGET" --include "$CFG_REL"

log "Restoring database: $DB_REL"
restic restore latest --target "$TARGET" --include "$DB_REL"

RESTORED_CFG="$TARGET/$CFG_REL"
RESTORED_DB="$TARGET/$DB_REL"

log "Checking restored files"
log "  CFG: $RESTORED_CFG"
log "  DB : $RESTORED_DB"

if [[ ! -s "$RESTORED_CFG" ]]; then
  log "ERROR: Restored config.php missing/empty"
  ls -lah "$TARGET" || true
  pushover "Nextcloud Restore-Test FAILED" "Restore of config.php failed or file is empty." 1
  exit 1
fi

if [[ ! -s "$RESTORED_DB" ]]; then
  log "ERROR: Restored db.sql missing/empty"
  pushover "Nextcloud Restore-Test FAILED" "Restore of db.sql failed or file is empty." 1
  exit 1
fi

# Real read test
head -n 3 "$RESTORED_CFG" >/dev/null
head -n 3 "$RESTORED_DB" >/dev/null
```

```
log "Running quick integrity check (read-data-subset=1%)"
restic check --read-data-subset=1% >/dev/null

rm -rf "$TARGET"

log "Restore test successful (snapshot=$SNAPSHOT_ID)"
log "========== RESTORE TEST OK =========="

pushover "Nextcloud Restore-Test OK" "Restore test successful. Snapshot: $SNAPSHOT_ID"
```

```
sudo chown root:root /root/nextcloud-restore-test.sh
sudo chmod 700 /root/nextcloud-restore-test.sh
```

Wöchentlicher Healthreport

Datei: /root/nextcloud-health-report.sh

```
#!/bin/bash
set -euo pipefail

# =========================
# Weekly Health Report
# =========================

ENV_FILE="/root/.nextcloud-backup.env"
if [[ -f "$ENV_FILE" ]]; then
  # shellcheck disable=SC1090
  source "$ENV_FILE"
else
  echo "ERROR: Env file not found: $ENV_FILE"
  exit 1
fi

: "${RESTIC_REPOSITORY:?RESTIC_REPOSITORY not set}"
: "${RESTIC_PASSWORD_FILE:?RESTIC_PASSWORD_FILE not set}"
: "${PUSHOVER_TOKEN:?PUSHOVER_TOKEN not set}"
: "${PUSHOVER_USER:?PUSHOVER_USER not set}"
```

```
export RESTIC_REPOSITORY
export RESTIC_PASSWORD_FILE

LOGFILE="/var/log/nextcloud-health-report.log"
exec >> "$LOGFILE" 2>&1

log() { echo "[$(date '+%F %T')] $*"; }

pushover() {
  local TITLE="$1"
  local MESSAGE="$2"
  local PRIORITY="${3:-0}"

  local DEVICE_ARG=()
  if [[ -n "${PUSHOVER_DEVICE:-}" ]]; then
    DEVICE_ARG=(-F "device=$PUSHOVER_DEVICE")
  fi

  curl -s \
    -F "token=$PUSHOVER_TOKEN" \
    -F "user=$PUSHOVER_USER" \
    "${DEVICE_ARG[@]}" \
    -F "title=$TITLE" \
    -F "priority=$PRIORITY" \
    -F "message=$MESSAGE" \
    https://api.pushover.net/1/messages.json >/dev/null || true
}

# Prevent concurrent runs with backup/restore-test
LOCKFILE="/var/lock/nextcloud-restic.lock"
exec 9>"$LOCKFILE"
flock -n 9 || { echo "Another backup/restore job is running. Exiting."; exit 0; }

CSVLOG="/var/log/nextcloud-backup.csv"
RESTORE_LOG="/var/log/nextcloud-restore-test.log"

log "========== WEEKLY HEALTH REPORT =========="

# ---- Last backup info from CSV ----
LAST_LINE=""
```

```
if [[ -f "$CSVLOG" ]]; then
  LAST_LINE="$(tail -n 1 "$CSVLOG" | tr -d '\r' || true)"
fi

LAST_STATUS="unknown"
LAST_DUR="?"
LAST_SNAP="?"
LAST_TS="?"

if [[ -n "${LAST_LINE:-}" && "$LAST_LINE" != timestamp,* ]]; then
  IFS=',' read -r LAST_TS _host LAST_STATUS LAST_DUR LAST_SNAP <<< "$LAST_LINE"
fi

# ---- Count backups in last 7 days (from restic, tag nextcloud) ----
WEEK_COUNT="?"
if restic snapshots --tag nextcloud --json >/dev/null 2>&1; then
  # Prefer JSON if possible, fallback to plain output count
  WEEK_COUNT="$(restic snapshots --tag nextcloud | grep -E '^[0-9a-f]{8,}' | wc -l | tr -d '
')"
fi

# ---- Latest snapshot time/id (from restic) ----
LATEST_SNAP_ID="?"
LATEST_SNAP_TIME="?"
if restic snapshots --tag nextcloud --latest 1 --json >/dev/null 2>&1; then
  # Extract short_id and time without jq
  LATEST_SNAP_ID="$(restic snapshots --tag nextcloud --latest 1 --json 2>/dev/null | grep -m1
'"short_id"' | sed -E 's/.*"short_id": ?"([^"]+)".*/\1/' || true)"
  LATEST_SNAP_TIME="$(restic snapshots --tag nextcloud --latest 1 --json 2>/dev/null | grep -
m1 '"time"' | sed -E 's/.*"time": ?"([^"]+)".*/\1/' || true)"
fi

# ---- Repo stats (if supported) ----
STATS_LINE="Stats: n/a"
if restic stats --mode=raw-data >/dev/null 2>&1; then
  # raw-data mode typically prints a compact table; keep it short
  STATS_LINE="$(restic stats --mode=raw-data 2>/dev/null | tail -n 5 | tr '\n' ' ' | sed -E
's/[[:space:]]+/ /g' | cut -c1-220)"
elif restic stats >/dev/null 2>&1; then
  STATS_LINE="$(restic stats 2>/dev/null | tail -n 8 | tr '\n' ' ' | sed -E 's/[[:space:]]+/
```

```
/g' | cut -c1-220)"
fi

# ---- Locks ----
LOCKS_MSG="Locks: none"
LOCKS_PRIORITY="0"
if restic list locks >/dev/null 2>&1; then
  LOCK_COUNT="$(restic list locks 2>/dev/null | grep -E '^[0-9a-f]{8,}' | wc -l | tr -d ' ' ||
true)"
  if [[ "${LOCK_COUNT:-0}" -gt 0 ]]; then
    LOCKS_MSG="Locks: ${LOCK_COUNT} (check 'restic unlock' if stale)"
    LOCKS_PRIORITY="1"
  fi
fi

# ---- Restore-test last result (best-effort from log) ----
RESTORE_STATUS="unknown"
if [[ -f "$RESTORE_LOG" ]]; then
  if tail -n 200 "$RESTORE_LOG" | grep -q "RESTORE TEST OK"; then
    RESTORE_STATUS="OK"
  elif tail -n 200 "$RESTORE_LOG" | grep -q "FAILED\|ERROR:"; then
    RESTORE_STATUS="FAIL"
  fi
fi

TITLE="Nextcloud Weekly Health"
PRIO="0"

# Escalate if last backup failed or locks exist
if [[ "${LAST_STATUS}" == "FAIL" ]]; then
  PRIO="1"
fi
if [[ "${LOCKS_PRIORITY}" == "1" ]]; then
  PRIO="1"
fi

MSG=$(
  cat <<EOF
Last backup: ${LAST_STATUS} (${LAST_DUR}s) @ ${LAST_TS} (snap ${LAST_SNAP})
Latest snapshot: ${LATEST_SNAP_ID} @ ${LATEST_SNAP_TIME}
```

```
Backups in repo: ${WEEK_COUNT}
Restore test: ${RESTORE_STATUS}
${LOCKS_MSG}
${STATS_LINE}
EOF
)

log "Sending weekly health report (priority=${PRIO})"
pushover "$TITLE" "$MSG" "$PRIO"

log "========== REPORT SENT =========="
```

```
sudo chown root:root /root/nextcloud-health-report.sh
sudo chmod 700 /root/nextcloud-health-report.sh
```

Cron einrichten

Wichtig ist, das Du als root eingeloggt sein musst.

```
crontab -e
```

Füge folgende Zeilen an das Ende ein.

Das Backup erfolgt jeden Tag um 3:30 Uhr und monatlich wird ein Restore-Test um 30:30 durchgeführt.

```
30 3 * * * /root/nextcloud-backup.sh
30 20 1 * * /root/nextcloud-restore-test.sh
0 9 * * 0 /root/nextcloud-health-report.sh
```

Rastic Kommandos

Alle Snapshots anzeigen lassen:

```
export RESTIC_REPOSITORY=/mnt/synology-backup/nextcloud
export RESTIC_PASSWORD_FILE=/root/.restic-password


restic snapshots
```

Spezifischen Teil aus einen bestimmten Snapshot wiederherstellen

Vorher habt Ihr die SnapshotID mit dem Kommando "Alle Snapshots anzeigen lassen" identifiziert!

```
restic restore SNAPSHOTID \
  --target / \
  --include /home/dockeruser/docker/nextcloud/nginx
```

Alle Backups in Rastic löschen

```
restic forget --prune --keep-last 0
```