

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"
        fi
    done
}

```

```

        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
```

File: /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
```

File: /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"
PRI0="0"

# Escalate if last backup failed or locks exist
if [[ "${LAST_STATUS}" == "FAIL" ]]; then
    PRI0="1"
fi
if [[ "${LOCKS_PRIORITY}" == "1" ]]; then
    PRI0="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=${PRI0})"
pushover "$TITLE" "$MSG" "$PRI0"

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
```

Revision #6

Created 2026-02-15 11:32:50 UTC by sylvio

Updated 2026-02-15 16:42:49 UTC by sylvio